WRITE YOUR OWN PROGRAM

# ADVANCED
# GRAPHICS

## HAUNTED HOUSE

# WRITE YOUR OWN PROGRAM

THIS NEW SERIES INTRODUCES THE ART OF PROGRAMMING YOUR COMPUTER. EACH BOOK SHOWS HOW TO STRUCTURE A PROGRAM INTO ROUTINES, AND AT THE SAME TIME EXPLAINS AND ANALYZES WHAT EXACTLY YOU ARE ASKING THE COMPUTER TO DO AND WHY.

AND THERE'S FUN TOO! EACH BOOK CONTAINS ONE OR MORE EXCITING AND ORIGINAL COMPUTER GAMES. THESE CAN BE ADAPTED AND EXTENDED TO DO BIGGER AND BETTER THINGS.

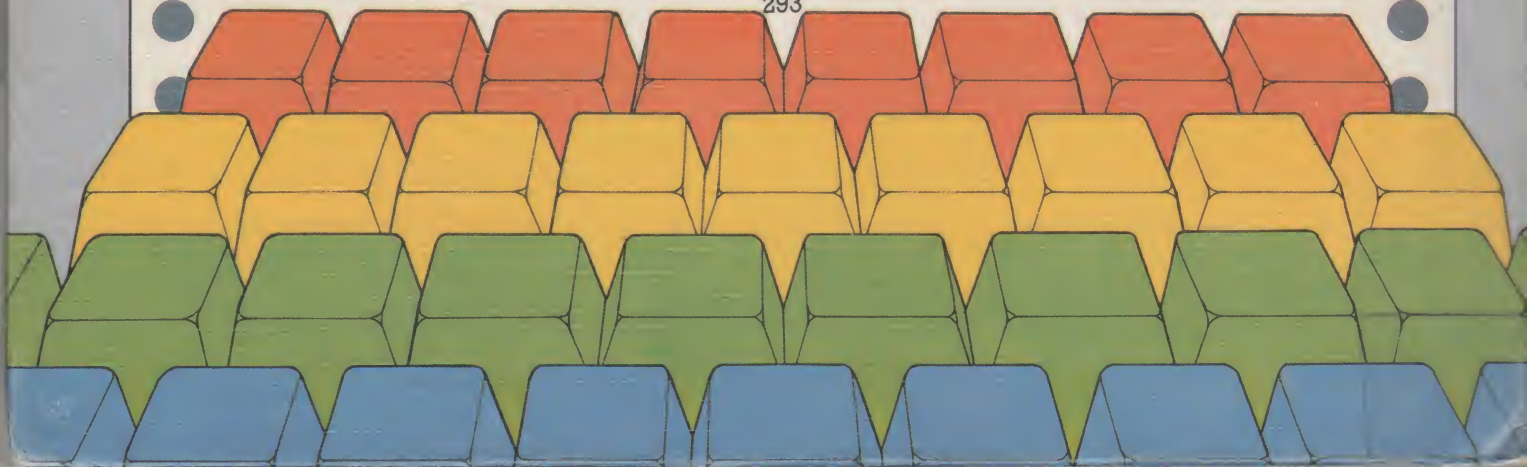THE BOOKS ARE SPECIFIC TO THE COMMODORE 64 AND APPLE IIe.

*WRITE YOUR OWN PROGRAM*

# COMPUTER ANIMATION

## HAUNTED HOUSE

**Marcus Milton**

# Foreword

There is much more to computer graphics than you may think. Instead of just playing or illustrating games, you can write a story, and make your own colorful cartoon film program using both static and animated graphics. The program is given for the Apple IIe and Commodore 64 computers, and tells the story of the Haunted House.

The program has been broken down into logical stages and a running text explains how each section works. Writing a program in logical stages not only makes it easier for you and other people to understand, it also reduces the possibility of errors. At various stages you can test the sections of the program you've keyed-in – even the smallest error can mean that the program doesn't work. If something does go wrong, check back through the listings very carefully. Make sure your program is exactly the same as is printed in the book.

The sinister goings-on around the haunted house use many useful computer graphics techniques. For example, by carefully designing some user-defined characters it is easy to get the smoothest animation from a BASIC program.

# Contents

```
3100    DATA 94,14599
3110    DATA 32,12,225,230,212,230,213,162,48,165,212
3120    DATA 5,213,208,2,162,0,142,35,96,165,161,73
3130    DATA 255,133,210,162,0,142,35,96,165,161,73
3140    DATA 173,0,192,165,160,73,255,133,211,73
3150    DATA 30,230,210,165,213,133,209,164,212,162,0
3160    DATA 212,198,213,208,26,230,211,208,212,202,208
3170    DATA 192,133,208,165,210,101,161,22,198
3180    DATA 173,16,192,197,208,208,227,76,242,226
3190    DATA 198,209,208,209,240,198,0,0
4000    REM READ IN THE INTERVAL TABLE FOR
4010    REM AN EQUAL TEMPERED SCALE
4020    READ NN: DIM NL%(NN),NH%(NN)
4030    FOR I = 1 TO NN: READ X
4040    LET T =    INT (X / 256)
4050    LET NL%(I) = X - 25
```

# APPLE Graphics

The Apple stores the details of the shapes that it is going to draw on the screen in special shape tables. That way a shape can be put into memory and called when it is needed. The shapes are numbered and at the beginning of the shape table there is an index so that the computer can find the particular shape you want drawn. The index contains the number of memory locations the beginning of the shape is away from the start of the table. Memory locations 232 and 233 tell the computer where the table is.

```
3000    READ SN: REM SHAPE TABLE SET UP
3010    POKE P,SN: POKE P + 1,0:ST = P + 2
3020    POKE 232, FN MOD(P): POKE 233, INT (P / 256)
3030    LET P = ST + 2 * SN
3040    FOR SS = 0 TO SN - 1:T = P - ST + 2
3050    LET A = ST + 2 * SS: POKE A, FN MOD(T)
3060    POKE A + 1, INT (T / 256):D$ = "":R = 0
3070    POKE P,25:P = P + 1: GOSUB 3150: NEXT : RETURN
```

To fully understand how Apple shape tables work you have to understand machine codes. But to make it easier a special BASIC routine has been written which translates simple direction commands into the right form. That routine is called by the **GOSUB** line 3070, after the overall format of the table has been set.

THE APPLES' SHAPE TABLES ARE MADE UP OF VECTORS. THESE ARE INSTRUCTIONS L MEANS GO LEFT, R MEANS GO RIGHT, U MEANS GO UP AND D MEANS GO DOWN. A P BEFORE THESE DIRECTIONS PRINTS ON THE SCREEN, AN M MOVES WITHOUT PRINTING. E MEANS END.

100 DATA PRUURDD6R5UR5D3RD3L5DL5U6LDDLUULE
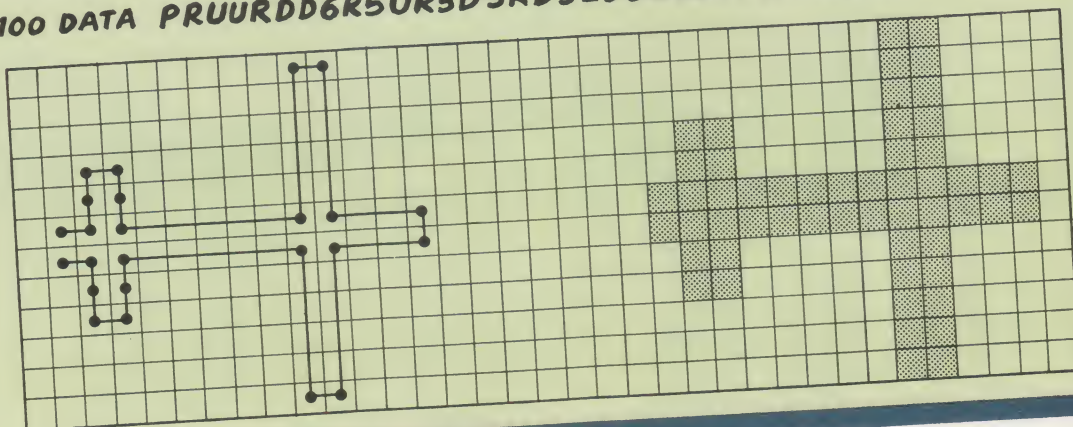
```
3100    LET X = X + B: FOR I = 1 TO R
3110    IF  PEEK (P - 1) > 8 OR X > 8 THEN 3130
3120    POKE P - 1,8 * X +  PEEK (P - 1): GOTO 3140
3130    POKE P,X:P = P + 1
3140    NEXT :R = 0
3150    IF D$ = "" THEN  READ D$: PRINT ".";
3160    LET A$ =  LEFT$ (D$,1)
3170    LET D$ =  MID$ (D$,2, LEN (D$) - 1)
3180    IF A$ = "R" THEN X = 1: GOTO 3100
3190    IF A$ = "D" THEN X = 2: GOTO 3100
3200 .  IF A$ = "L" THEN X = 3: GOTO 3100
3210    IF A$ = "U" THEN X = 128: GOTO 3100
3220    IF A$ = "M" THEN B = 0: GOTO 3150
3230    IF A$ = "F" THEN B = 4: GOTO 3150
3240    IF A$ = " " THEN 3150
3250    IF A$ < "0" OR A$ > "9" THEN 3270
3260    LET R = 10 * R +  VAL (A$): GOTO 3150
3270    POKE P,X: POKE P + 1,0:P = P + 2: RETURN
```

This is the routine that translates the simple direction
instructions that are going to be given in the drawing data
into the complicated numbers that the shape table requires.
In line 3150, the **DATA** is **READ** in as string and **PRINT**s a dot
on the screen to show that the computer is working during
this long procedure. Lines 3160 and 3170 pick off the first
number or letter off the string. Lines 3180 through 3250 look
for the direction letters we're using or numbers. Lines 3100
through 3140 do the complicated sums that work out the
shape table numbers and **POKE** them into the right place in
the shape table.

YOU CAN MAKE AN AIRPLANE FLY
ACROSS YOUR SCREEN BY TYPING IN
LINES 3000 TO 3270 WITH THIS SHORT
PROGRAM AND LINE 100 OPPOSITE.

```
10  HGR : HCOLOR = 3
15  P = 24576 : ROT = 0 : SCALE = 1
20  GOSUB 3000
25  X DRAW 1 AT 10,20
30  FOR X = 11 TO 270
35  X DRAW 1 AT X-1, 20
40  X DRAW 1 AT X,20
45  NEXT
50  X DRAW 1 AT 270,20
55  TEXT : END
60  DATA 1
```

# COMMODORE Graphics

To draw the haunted house, you have to use what is called
"high resolution graphics" where each little dot on the TV
screen is controlled individually. But Commodore BASIC has
no simple command for drawing a line in high resolution
graphics. So the program below loads a machine code
routine which will do the job. You have to type this program
in and **RUN** it before the main program will work.

```
5 REM ***     STARTER PROGRAM       ***
10 FOR I=0 TO 214: READ D: CS=CS+D
20 POKE 49152+I,D: NEXT I
30 IF CS<>29713 THEN PRINT "DATA ERROR!":END
40 POKE642,64:SYS64766
100 DATA0,32,253,174,32,158,183,134,253,32,253,174
110 DATA32,158,183,138,72,32,253,174,32,158,183,138
120 DATA41,7,133,254,138,74,74,74,133,251,10,10
130 DATA101,251,160,6,162,0,134,252,42,38,252,136
140 DATA208,250,101,254,133,251,165,253,41,252,10,144
150 DATA3,230,252,24,101,251,133,251,165,252,105,32
160 DATA133,252,32,253,174,32,158,183,138,141,0,192
170 DATA160,3,10,10,13,0,192,136,208,248,141,0
180 DATA192,165,253,74,74,133,255,104,133,254,74,74
190 DATA56,229,255,240,52,133,255,32,178,192,32,149
200 DATA192,169,8,24,101,251,133,251,165,252,105,0
210 DATA133,252,198,255,240,8,173,0,192,145,251,76
220 DATA121,192,32,197,192,72,73,255,49,251,145,251
230 DATA104,45,0,192,17,251,145,251,96,32,178,192
240 DATA133,255,32,197,192,37,255,76,149,192,169,255
250 DATA72,165,253,41,3,240,8,170,104,74,74,202
260 DATA208,251,96,104,96,169,255,72,169,252,5,254
270 DATA170,104,232,208,1,96,10,10,76,206,192
```

The numbers in the **DATA** lines are the machine code and
the loop between lines 10 and 20 **READ**s these numbers and
**POKE**s them into the computer's memory. But to check that
you do not make a mistake when you are keying them in, line
10 also adds the numbers up. The total is checked in line 30
which then tells you if you have made a mistake. Line 40
makes sure that when the main program is loaded it goes
into an area of memory where it won't interfere with the
machine code. The machine code is called by the command
**SYS MC**. This command is followed by four numbers which
tell the machine code where the ends of the line are, how far
down the screen and what color it is.

```
13000 REM ***          DRAW BOX          ***
13010 FOR I=Y1 TO Y2: FOR J=X1 TO X2
13020 POKE SC+I*40+J,134: POKE CO+I*40+J,C
13030 NEXT J,I
13040 FOR Y=Y1*8 TO Y2*8+7
13050 SYS MC,X1*4,X2*4+3,Y,3
13060 NEXT Y
13070 IF NS=1 THEN RETURN
13080 FOR Y=Y2*8+7 TO Y1*8 STEP-2
13090 SYS MC,X1*4,X2*4+3,Y,1
13100 NEXT Y
13110 RETURN
```

This subroutine uses the machine code put into memory by
the starter program to draw a box. So **RUN** the starter
program, type in this routine, then test it using the test
program below. When it is working, delete the test lines. **Y1**,
**Y2**, **X1** and **X2** are the row and column numbers that form the
sides of the box. The loop between lines 13010 and 13030
**POKE**s the color memory of each character square in the
box. And the loop between 13080 and 13100 uses **SYS MC** to
call the machine code routine which fills in the box a line at a
time with that color. The loop between 13080 and 13100
works back up the box, using **SYS MC** to change the color of
every other line to orange to give stripes. If **NS** is fixed at 1
outside the program the machine code routine does not
draw in the stripes.

BEFORE YOU GO ANY FARTHER
IT IS BEST TO TEST THAT YOUR
MACHINE CODE ROUTINE AND
BLOCK-DRAWING ROUTINE IS
WORKING. THIS LITTLE TEST
PROGRAM WILL DO JUST THAT.

```
1 B=8192:MC=49153:BM=53265:CH=53270:SC=1024:CO=55296
2 MP=53272:Y1=0:Y2=24:X1=0:X2=39
3 POKE MP,PEEK(MP)OR8:POKE(BM),PEEK(BM)OR32
4 POKE CH,PEEK(CH)OR16:C=0:GOSUB 13000
5 X1=15:X2=25:Y1=7:Y2=17:C=C+1:GOSUB13000:GOTO 5
```

# The storyboard

The storyline of the cartoon must be worked out in detail before the main program is written. First the scene is set, then the mysterious castle appears in the middle of nowhere, surrounded by a spooky forest. Night falls. A thunderous lightning flash wakes the bats in the belfry and an evil old witch takes to the air on her broomstick. As she flies off into the night accompanied by more lightning flashes. Then suddenly the old house catches fire. The flames leap higher and higher. The house is burnt to the ground. And, as an eerie conclusion to this macabre tale, a huge ghostly skull appears in the sky, hovering over the ashes of the house. When working on a cartoon tale like this one, it is a good idea to tackle the task like a professional movie maker. Work out a detailed storyboard, like the one below, that breaks the plot down into a series of separate events.
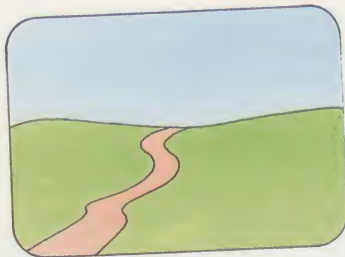


BACKGROUND
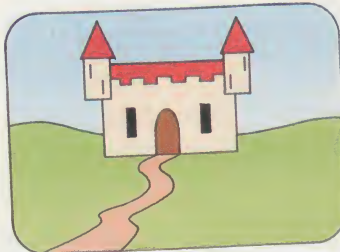
HAUNTED HOUSE

HAUNTED FOREST

NIGHT FALLS

BAT AND WITCH

LIGHTNING STRIKE

THE BLAZING HOUSE

THE HOUSE COLLAPSES

A SKULL APPEARS

# THE CONTROL PROGRAM AND INITIALIZATION

A control program is one that calls a series of subroutines in the right order. Here the control program corresponds to the storyboard and the subroutines correspond to each of the separate scenes.
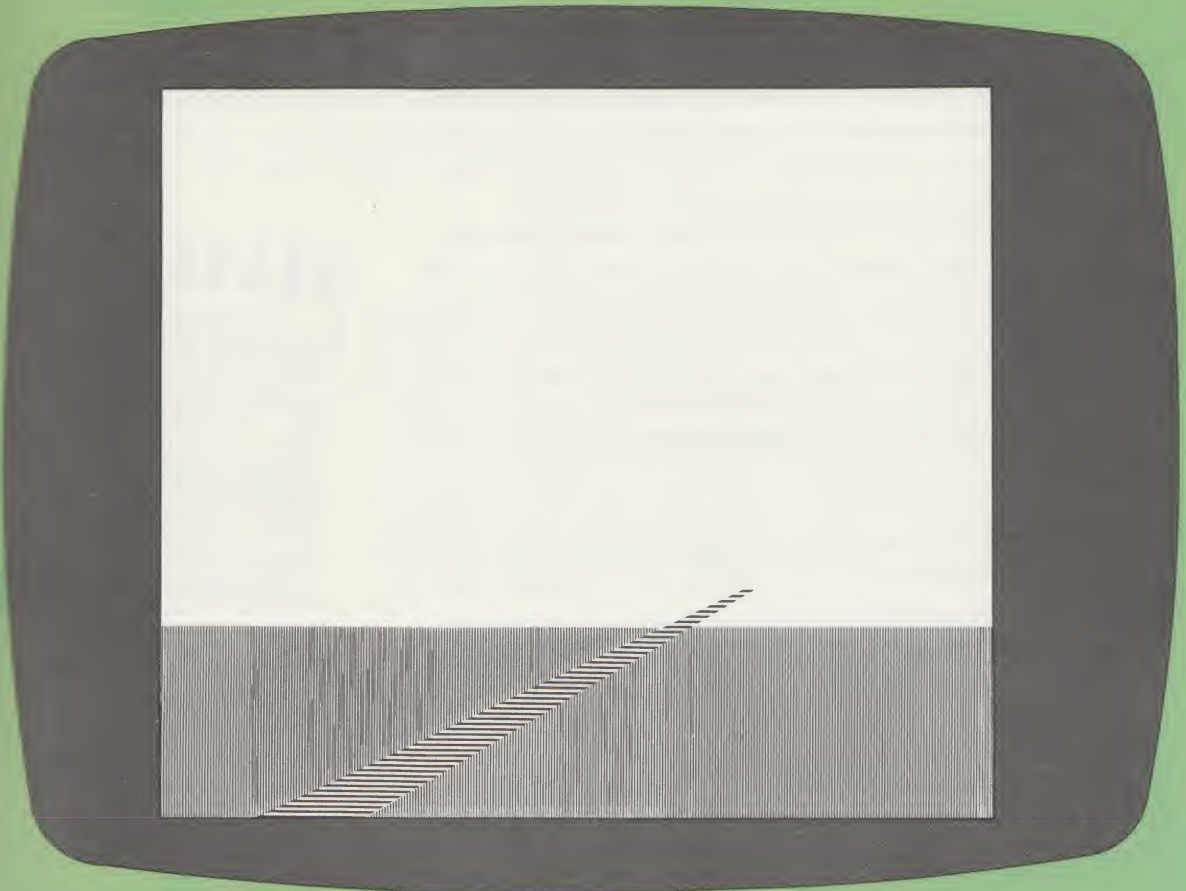
The control program follows the storyboard, with each subroutine being called in turn. At first they simply set up the machine and draw up each element of the scenery. But when the action starts things get a bit more complicated.

```
1    REM HAUNTED HOUSE
5    DEF  FN MOD(X) = 256 * (X / 256 -   INT (X / 256))
100    GOSUB 1000: REM INITIALIZATIONS
110    GOSUB 6000: REM BACKGROUND & CASTLE
120    GOSUB 6500: REM TREES
130    GOSUB 5000: REM NIGHTFALL
140    GOSUB 2400: REM LIGHTNING
150    FOR TI = 1 TO 250
160    IF TI = 10 THEN  GOSUB 2100: REM START WITCH
170    IF TI > 10 THEN  GOSUB 2150: REM MOVE WITCH
180    IF TI = 30 THEN  GOSUB 2200: REM START BAT
190    IF TI > 30 THEN  GOSUB 2230: REM MOVE BAT
200    IF  RND (1) < .07 THEN  GOSUB 2400: REM LIGHTNING
210    NEXT TI: GOSUB 2300: REM REMOVE BAT
220    GOSUB 2400: GOSUB 2400: REM STORM
230    GOSUB 2500: REM FIRE
240    FOR X = 1 TO 1000: NEXT : GOSUB 2700: REM SKULL
250    TEXT : END
```

Because of the special way machine code numbers have to be handled, line 5 starts off by defining the function **FN MOD (X)** which will help simplify the math. Next the initialization subroutine is called, which tells the computer all the things that it will need to know to run the program. Then the routines that draw up the background and castle, the trees, make night fall and make the lightning flash are all called. **TI** is going to be used as a clock to control the animation. The routines that draw up the witch, make her move and draw up the bat and make it move take their cue from TI. In line 200 the Apple's dice-rolling function **RND** is used to call the lightning program and give lightning flashes at random times.

Once the animation loop has ended, the computer goes on with the rest of line 210 and removes the bat. Line 220 calls the lightning routine twice in quick succession to simulate a storm. In line 230 fire breaks out – or, at least, it's set by calling the routine at line 2500.

Once the castle has burned down to the ground, there is a thousand time period pause to build up dramatic tension before the routine that puts up the eerie skull is called. Line 250 sets the Apple back to normal **TEXT** mode before **END**ing the program.
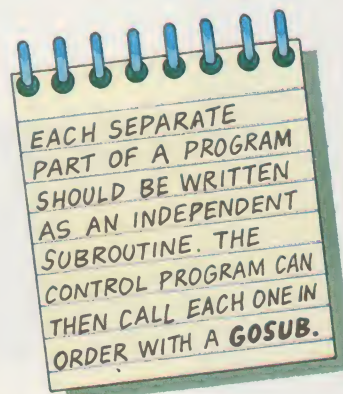
EACH SEPARATE PART OF A PROGRAM SHOULD BE WRITTEN AS AN INDEPENDENT SUBROUTINE. THE CONTROL PROGRAM CAN THEN CALL EACH ONE IN ORDER WITH A **GOSUB.**

```
1000    REM INITIALIZE PEEK AND POKE ADDRESSES
1010    LET P1GE = 49236:P2GE = 49237
1020    LET FULLSCREEN = 49234:GPAGE = 230
1030    LET CLICK = 49200:P = 24576
1040    GOSUB 3000: REM SET UP SHAPE TABLE
1050    GOSUB 5200: REM PUT MACHINE CODE IN
1060    HGR :X =  PEEK (FULLSCREEN)
1070    SCALE= 1: ROT= 0: RETURN
```

When initializing any program it is best to give it all the important memory addresses names that will be easy to remember. **P1GE** and **P2GE** are the addresses that switch on graphics screens 1 and 2. **FULLSCREEN** is the address which switches the Apple to full graphics screen, rather than the one that mixes graphics with text. **CLICK** is the address that switches on the speaker. **P** is the pointer to the shape table. The address given here is the starting address of the shape table, but **P** will be updated as the shapes are loaded in. Line 1040 calls the subroutine that sets up the shape table, so now the computer has all the shapes at its fingertips. Next a subroutine is called that loads in a machine code routine. This routine copies graphics screen one onto the graphics screen two, changing all the colors at the same time. **HGR** puts the Apple into high-resolution graphics mode and **PEEK**ing **FULLSCREEN** selects the full graphics screen. **SCALE** and **ROT** say that the shapes should be drawn at the same size and at the same angle as they were given.

## SHAPE TABLE

THE STARTING ADDRESS OF THE SHAPE TABLE ( BYTE + $\phi$) IS STORED AT ADDRESSES 232 AND 233.

BYTE + 0 CONTAINS THE NUMBER OF SHAPES. IN THIS EXAMPLE THERE ARE 2.

STARTING AT BYTE + 2 THERE IS A LOW BYTE/ HIGH BYTE INDEX TO THE SHAPE DEFINITIONS RELATIVE TO BYTE + 0.

AFTER THE INDEX ARE THE SHAPE DEFINITIONS THEMSELVES.

( ALL SHAPE DEFINITIONS END WITH A ZERO.)

| | | |
|---|---|---|
| BYTE + 0 | NUMBER OF SHAPES | 2 |
| BYTE + 1 | UNUSED | ($\phi$) |
| BYTE + 2 | ADDRESS SHAPE DEF. 1 ( LOW ) | 6 |
| BYTE + 3 | ADDRESS SHAPE DEF. 1 ( HIGH ) | $\phi$ |
| BYTE + 4 | ADDRESS SHAPE DEF. 2.( LOW ) | 38 |
| BYTE + 5 | ADDRESS SHAPE DEF 2 ( HIGH ) | $\phi$ |
| BYTE + 6 | SHAPE DEF. 1   START | |
| | | |
| BYTE + 37 | SHAPE DEF. 1   END | $\phi$ |
| BYTE + 38 | SHAPE DEF. 2   START | |
| | | |
| BYTE + 69 | SHAPE DEF. 2   END | $\phi$ |

15

The structure of this control program follows the storyboard exactly. It calls the routines one after another. The only routines that are called more than once are those that move the witch and the bat and make the lightning. This is to make the witch and bat move continuously and the lightning flash repeatedly.

```
5 REM ***     ===HAUNTED HOUSE===     ***
10 GOSUB 14000:REM INITIALIZE
20 GOSUB 1000 :REM BACKGROUND
30 GOSUB 2000 :REM CASTLE
40 GOSUB 8000 :REM TREES
50 GOSUB 9000 :REM NIGHT
60 POKE V+21,243:DX=-1:DY=-1
70 FOR WX=10 TO 346 STEP 2:LI=RND(0)*20
80 IF LI>19 THEN GOSUB 10000:REM LIGHTNING
90 IF WX>255 THEN RS=255:POKE V+16,3
100 POKE V+0,WX-RS:REM WITCH
110 GOSUB 10040:REM BAT
120 NEXT WX: POKE V+21,240
130 GOSUB 11000:REM BURN
140 GOSUB 12000:REM SKULL
150 END
```

This program calls a number of subroutines – the name of each subroutine is given in the **REM** line following the **GOSUB** call. The Commodore has a special graphics facility called sprites. These allow you to move areas of animation across the screen easily. But they are controlled by **POKE**ing a special area of memory. The **POKE** in line 60 switches on the sprites which contain the bat and the witch. **DX** and **DY** fix the direction in which the bat starts moving. Setting them both to −1 means that the bat starts out moving from right to left and up the screen. **WX** controls the movement of the witch. She starts off in column 10 and moves across the screen two steps at a time until she disappears off the right-hand side. The lightning is made to flash randomly by using the dice-throwing instruction **RND**. Roughly once in every 20 throws **RND (0)\*20** will come up with a number bigger than 19 and the lightning subroutine at 1000 will be called.

*RND (0) GIVES A RANDOM DECIMAL NUMBER BETWEEN 0 AND 1 SO RND (0)\*20 GIVES A RANDOM NUMBER BETWEEN 0 AND 20*

One of the problems with using sprites is that you have to deal directly with the Commodore's memory locations. But each memory location can only store a number between 0 and 255. But there are more than 255 columns across the screen. So when the witch reaches column 255, adjustments have to be made. These are done by lines 90 and 100. Line 120 switches the bat and the witch sprites off again when you've finished with them.
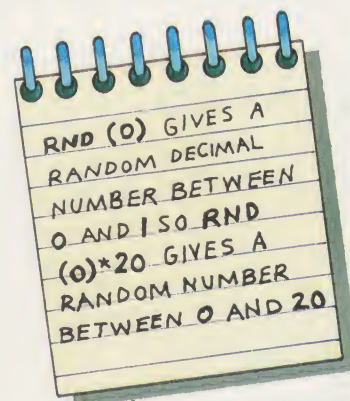
16

```
14000 REM ***          INITIALIZE        ***
14010 PRINT"◼":POKE53280,11:POKE53281,14
14020 POKE53270,PEEK(53270)OR16:REM MULTICOLOUR MODE
14030 POKE53272,PEEK(53272)OR8:REM PUT SCREEN AT 8192
14040 POKE53265,PEEK(53265)OR32:REM SWITCH TO BITMAP
14050 DIM TR(33,15),DO(16,8),MO(37,10),SK(28,14)
14060 FORI=0TO20:FORJ=0TO15:READTR(I,J):NEXTJ,I
14070 FORI=0TO16:FORJ=0TO7:READMO(I,J):NEXTJ,I
14080 FORI=0TO14:FORJ=0TO9:READSK(I,J):NEXTJ,I
14090 V=53248:SC=1024:CO=55296:MC=49153:LI=0
```

To initialize this program we have to put the computer into
the right mode to give high resolution graphics, **READ** in the
**DATA** for all the things that have to been drawn on the
screen, and give easy-to-remember names to important
memory locations. Line 14010 clears the screen, sets the
background color to light blue and sets the border color to
gray. Lines 14020 through 14040 put the computer into high
resolution graphics mode. And lines 14060 through 14080
**READ** the **DATA** for the trees, the moon and the skull into the
arrays **TR**, **MO** and **SK**. Line 14090 gives names to the
important memory locations. **V** is the start address of the
sprite memory. **SC** and **CO** are start addresses of the screen
and color memories. And **MC** is the address of the special
machine code high-resolution line-drawing program. Sprites
are easy to use, so the windows of the castle will also be
made from sprites. Lines 14100 through 14130 set their X and
Y positions. Lines 14140 and 14150 set the starting positions
of the witch and the bat. Lines 14160 through to 14190 set the
colors and tells each sprite where to find its **DATA**. That
**DATA** is then **READ** and **POKE**d into memory by line 14200.
Line 14210 fills the screen with the sky color by using the
machine code program.

```
14100 POKEV+8,120:POKEV+ 9,107:REM WINDOWS
14110 POKEV+10,232:POKEV+11,107
14120 POKEV+12,142:POKEV+13,147
14130 POKEV+14,210:POKEV+15,147
14140 POKEV+00,10 :POKEV+01,60 :REM WITCH
14150 POKEV+02,50 :POKEV+03,80 :REM BAT
14160 FORI=0TO7:POKEV+39+I,1:NEXT:REM COLOURS
14170 FORI=4TO7:POKE2040+I,32:NEXT:REM POINTERS
14180 POKE2040,34:POKE2041,35:POKEV+16,2
14190 POKEV+37,10:POKEV+38,8:POKEV+28,240:REM MC MODE
14200 FORI=0TO319:READ D:POKE2048+I,D:NEXT
14210 FOR Y=0TO199:SYS MC,0,159,Y,0:NEXT
14220 RETURN
```

Although there are 10 trees on the screen, only one lot of
**DATA** is needed. All the trees look the same so the same
**DATA** can be used each time a tree is drawn. The numbers
in the **DATA** here tell our machine code program which
color to use. A **4**, though, is not a color recognized by that
routine. So if a 4 is found no action is taken. In the tree **DATA**,
you'll see that there are only 4s and 2s. The 4s do nothing and
the 2s draw a point on the screen. If you look at the pattern of
the 2s, you'll see that they draw out the shape of a weird,
gnarled tree.

```
20000 REM          DATA FOR TREE
20010 DATA4,2,4,4,4,4,4,2,4,4,2,4,2,4,4,4
20020 DATA2,2,4,4,2,2,4,4,2,4,4,2,4,4,4,4
20030 DATA4,4,2,4,4,2,4,4,2,4,4,2,4,4,4,4
20040 DATA4,4,2,4,4,4,2,4,2,4,4,2,4,2,4,4
20050 DATA4,4,4,2,4,2,4,2,2,2,4,4,2,4,2,4
20060 DATA4,4,4,4,2,4,4,4,2,4,4,2,2,4,4,4
20070 DATA4,4,4,4,2,4,4,4,2,4,4,2,2,4,4,4
20080 DATA4,4,4,2,2,4,4,4,2,4,2,2,4,4,4,4
20090 DATA4,4,2,4,2,2,2,2,2,4,2,4,4,4,2,4
20100 DATA4,4,4,2,4,4,2,2,2,2,4,2,4,4,4,4
20110 DATA4,4,4,2,4,4,2,2,2,2,4,2,4,4,4,4
20120 DATA4,4,4,4,4,4,2,2,2,2,4,4,4,4,4,4
20130 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4
20140 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4
20150 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4
20160 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4
20170 DATA4,4,4,4,4,4,2,2,2,2,4,4,4,4,4,4
20180 DATA4,4,4,4,4,4,2,2,2,2,4,4,4,4,4,4
20190 DATA4,4,4,4,4,2,2,2,2,2,4,4,4,4,4,4
20200 DATA4,4,4,4,2,2,2,2,2,2,2,4,4,4,4,4
20210 DATA4,2,2,2,4,2,2,2,2,2,2,2,2,2,4,4
```

# DAYTIME

The background scenery, the haunted castle and the spooky
forest are drawn up on the screen during the daytime.
Everything is now ready for the action, which only begins
once the sun has gone down.
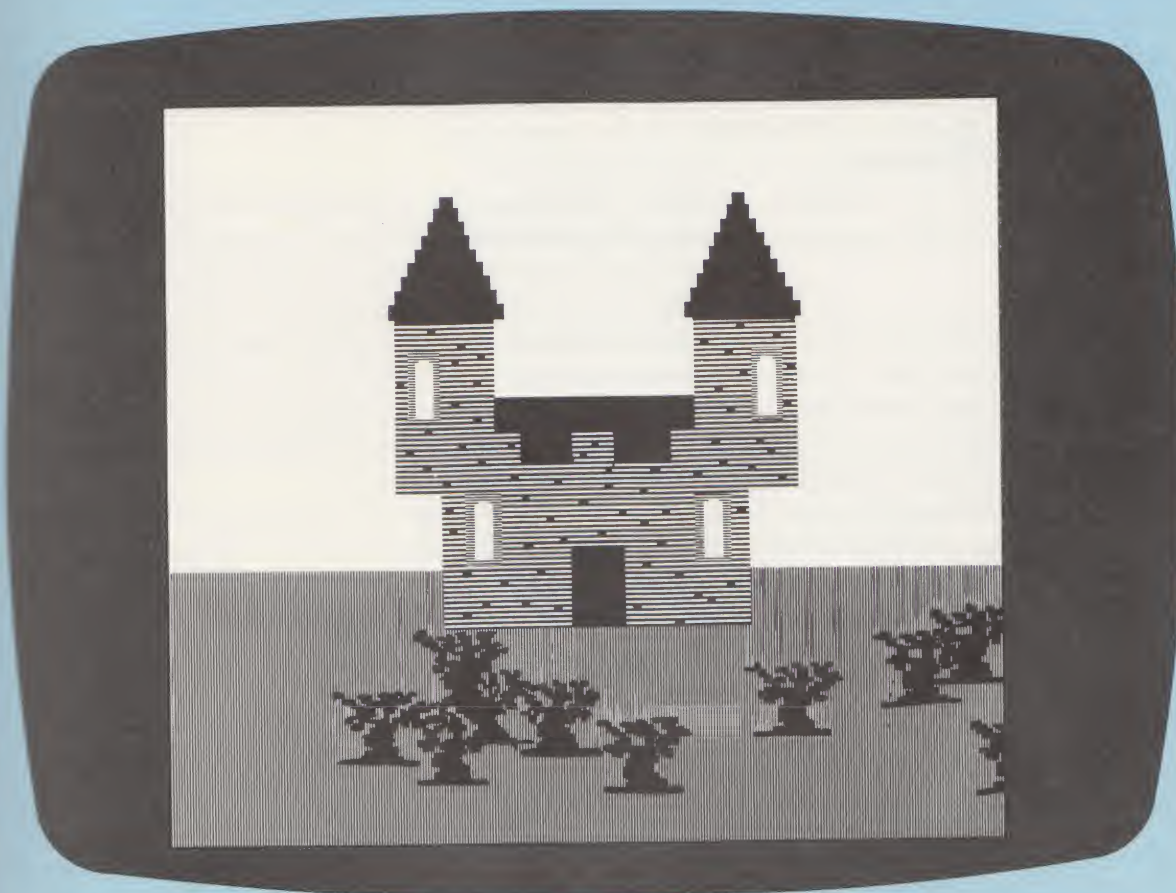
PLAN YOUR GRAPHICS CAREFULLY BEFORE YOU START WRITING YOUR PROGRAM.

```
7000    REM SKY,GRASS & PATH.
7010    DATA ORIGIN,0,0,0,0,0,0,0,0
7020    DATA BLOCK,  0,  0,279,135,  0,  0,6,6
7030    DATA BLOCK,  0,136,279,191,  0,  0,1,1
7040    DATA TRIANGLE, 50,191,180,126, 75,191,0,3
```

Not only does this program give you a BASIC routine to
simplify the shape data, it also gives you one that simplifies
the input of graphics data. You can use these routines – or
routines like them – to simplify drawing on the screen in your
own programs. This routine gives you new instructions that
are not normally available in BASIC. Two main instructions
are given here: **BLOCK** and **TRIANGLE**. All the instructions
are followed by eight numbers. The first four specify the
sides of a block or two corners of a triangle. The next two fix
the third corner of the triangle and are zero with the **BLOCK**
instruction.

The last two are the colors that the block or triangle is to be
drawn in. There are two colors because the block and
triangles are going to be striped. Added to that, there are
two other new instructions which work with these two.
**ORIGIN** fixes the point the sides of the block or the points of
the triangle are measured from. This point is specified by the
first two numbers, the rest are zero. And **STOP** switches off
the special graphics routine. The sky and the grass are
drawn as two blocks, the castle as three, one for the main
body, and two turrets. The path is a long, thin triangle.

```
7050    REM CASTLE DATA
7060    DATA ORIGIN, 63, 30,0,0,0,0,0,0
7070    DATA BLOCK, 28, 60,118,120,  0,  0,1,2
7080    DATA BLOCK, 14, 40, 41, 80,  0,  0,1,2
7090    DATA BLOCK,105, 40,132, 80,  0,  0,1,2
```

```
6000    REM GRAPHIC DATA INTERPRETER
6010    READ R$,X(1),Y(1),X(2),Y(2),X(3),Y(3),C(0),C(1)
6020    IF R$ = "ORIGIN" THEN OX = X(1):OY = Y(1)
6030    FOR I = 1 TO 3:X(I) = X(I) + OX
6040    LET Y(I) = Y(I) + OY: NEXT I
6050    IF R$ = "BLOCK" THEN  GOSUB 6100
6060    IF R$ = "TRIANGLE" THEN  GOSUB 6200
6070    IF R$ = "STOP" THEN  RETURN
6080    GOTO 6010
```

Line 6010 **READ**s in a string, **R$**, then the numbers that follow
it. Line 6020 looks for the word "ORIGIN" and sets the
coordinates of the origin as **OX** and **OY**. Line 6030 through
6040 add the coordinates of the origin into any coordinates
given after the other new instructions. Line 6050 looks for the
word "BLOCK" and goes to the block-drawing subroutine if it
finds it. And line 6060 looks for "TRIANGLE". If line 6070 finds
"STOP" the computer returns to the control routine.

In the block routine, the sum $T = 1 - T$ is used to change the
color each time the computer goes round a loop, so each line
is drawn in a different color. **HCOLOR** – the high-resolution
color – is set to **C(T)**. And **HPLOT** draws a line across the
screen at the height given by **Y**. So the block routine draws
in a block a line at a time, alternating the color of the lines to
give stripes.

WHEN $T=0$, $T=1-T$
MAKES $T$ EQUAL
TO 1. AND WHEN
$T=1$, $T=1-T$
MAKES IT EQUAL
TO 0 AGAIN.

The triangle routine works in much the same way, but each
line is a different length. This length is worked out by lines
6210 and 6220. Each line across the triangle gets bigger as
you go down it. How much bigger each line is, is worked out
by dividing the distance between the points of the triangle
by the height. This length is then multiplied up as the lines
are drawn down the triangle. The color of the lines is
alternated using the sum $T = 1 - T$ again.

```
6100    LET T = 1: REM FILL BLOCK  IN 2 COLOURS
6120    FOR I = Y(1) TO Y(2):T = 1 - T
6130    HCOLOR= C(T): HPLOT X(1),I TO X(2),I
6140    NEXT : RETURN
6200    REM FILL TRIANGLE APEX (X2,Y2)
6210    LET H = Y(1) - Y(2):K1 = (X(2) - X(1)) / H
6220    LET K2 = (X(3) - X(2)) / H:T = 1
6230    FOR I = 0 TO H:T = 1 - T: HCOLOR= C(T)
6240    HPLOT X(1) + K1 * I,Y(1) - I
6245    HPLOT  TO X(3) - K2 * I,Y(3) - I
6250    NEXT : RETURN
```

```
7100    REM ROOFS ON TURRETS
7110    DATA TRIANGLE, 10, 39, 28,  0, 44, 39,5,5
7120    DATA TRIANGLE,103, 39,118,  0,136, 39,5,5
7130    REM BATTLEMENTS
7140    DATA BLOCK, 42, 45,103, 59,  0,  0,5,5
7150    DATA BLOCK, 49, 60, 55, 65,  0,  0,5,5
7160    DATA BLOCK, 63, 60, 69, 65,  0,  0,5,5
7170    DATA  BLOCK, 77, 60, 83, 65,  0,  0,5,5
7180    DATA BLOCK, 91, 60, 97, 65,  0,  0,5,5
```

The roofs are put on the turrets by drawing two large triangles. These are not striped though. But the triangle routine draws lines alternately in the two colors specified. So the two colors are simply set to the same value. Both are color **5**. The origin for this section was fixed in the last bunch of **DATA**, the **DATA** for the castle.

The battlements of the castle are drawn in by extending the roof color into the body of the castle. This is done by using the **BLOCK** command to draw five small squares in the roof color over the edge of the block that has already been drawn to make the castle's walls. The block is not striped so the both colors are the same again – both color **5**, the same as the roof. You will notice again that the fifth and sixth numbers that follow the **BLOCK** command are zero.

The castle's windows and doors are drawn in using the **BLOCK** command too. This **DATA** is still working from the origin given in the castle **DATA**. The roofs on the turrets, the battlements and the windows and doors are all drawn over the original castle, so it makes sense to use the same origin. This time the blocks are striped. The windows and doors are drawn in alternate lines of color **0**, black, and color **3**, white. The **STOP** command that switches off the graphics routine has only zeros following it. It may seem unnecessary to have all these zeros, but the routine that **READ**s the **DATA** is expecting eight numbers.

```
7190    REM WINDOWS & DOOR
7200    DATA BLOCK, 21, 43, 27, 51,  0,  0,0,3
7210    DATA BLOCK,119, 43,125, 51,  0,  0,0,3
7220    DATA BLOCK, 28, 60, 34, 70,  0,  0,0,3
7230    DATA BLOCK,112, 60,118, 70,  0,  0,0,3
7240    DATA BLOCK, 35, 90, 48,102,  0,  0,0,3
7250    DATA BLOCK, 98, 90,111,102,  0,  0,0,3
7260    DATA BLOCK, 63,105, 83,120,  0,  0,0,3
7270    DATA STOP,0,0,0,0,0,0,0,0
```

```
6500     HCOLOR= 3: REM TREES
6510     FOR T = 1 TO 20
6520     LET X =  RND (1) * 250:Y =  RND (1) * 26
6530     DRAW 1 AT X + 25,191 - Y: NEXT : RETURN
```

This routine draws twenty trees – one in each pass of the
**FOR . . . NEXT** loop between lines 6510 and 6530. Line 6520
rolls the computer's dice function, **RND**, to position the trees
in random places. The **DRAW 1** in line 6530 draws the first
shape from the shape table on the screen. The first shape is,
of course, the tree. The position is given by the expressions
that follow **AT** in line 6530.

```
4000     REM 9 GRAPHIC ELEMENTS
4010     DATA  9
4100     REM   TREE
4110     DATA P2LUD 9LM3L  P2LMU  2RP7R  MRUP7  LMU2R  P6RU5
4120     DATA LU4RM  ULP3L  MURP3  RMRUP  4LU4R  U4LML  UP4RU
4130     DATA 4LU4R  U4LML  UP7RM  5RUP2  LM2LP  14LD3  LDL3D
4140     DATA MR8UP  DR2D6  RM3RP  9RMRU  P11LU  ULM3L  P2LMU
4150     DATA RPRU3  LULM3  RP3RU  2LU2L  U2LUM  4U2RP  2RDRD
4160     DATA RLDRD  RLDRD  RLDRD  RLDRD  RDLDR  MDD3R  P4RM2
4170     DATA RP3RU  L3RUR  URDDM  M10U3  LPLDL  D6LDR  3URUR
4180     DATA M7LPL  3D2R2  D2RU2  DLDRD  2LDRD  LDRE
```

There are nine shapes in the shape table. This is set by the 9
in the **DATA** in line 4010 which is read in at the start of the
table. The code here is simple to follow. **P** means print, **L**
means left, **R** means right, **U** means up, **D** means down and **M**
means move. Directions following a **P** fill in the next point in
that direction with color. But directions following an **M** move
the point in that direction without coloring it. A number
before a direction tells the computer to move or print that
many points in that direction. If you draw out these
movements on a piece of graph paper you'll see the
shape of a tree emerging.
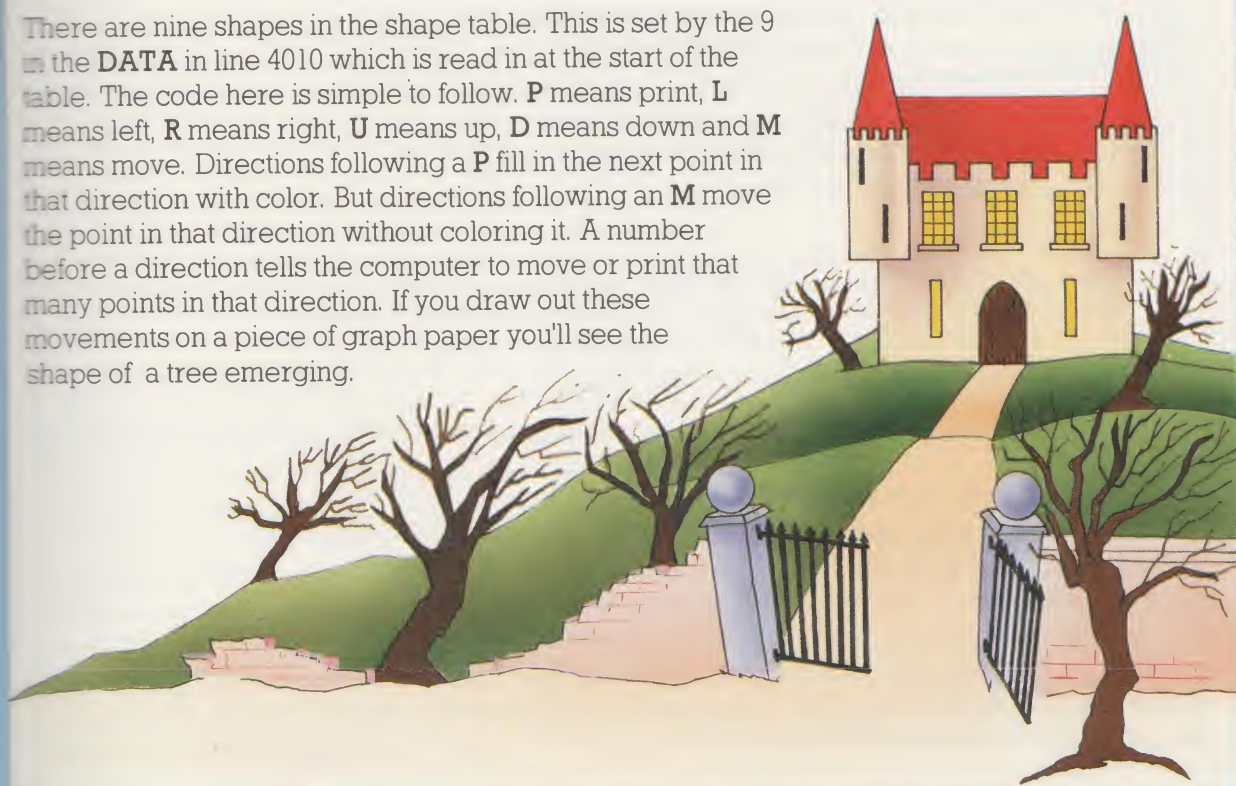
```
1000 REM ***          BACKGROUND          ***
1010 FOR I=0 TO 39: POKE 55856+I,11: NEXT
1020 FOR I=15 TO 24: FOR J=0 TO 39
1030 POKE CO+I*40+J,5: POKE SC+I*40+J,121
1040 NEXT J,I
1050 SYS MC,0,159,119,3
1060 FOR Y=120 TO 199: SYS MC,0,159,Y,3
1070 NEXT Y: RETURN
```

The background routine does not draw up much of the background, but it does set up areas of the screen and color memories so that the routines that do the drawing later will know which color to use. Line 1010 **POKE**s the color memory along the horizon with gray. And lines 1020 through 1040 **POKE** the color memory of the bottom of the screen with green for the grass and the screen memory with cyan and brown, ready to draw the trees. Line 1050 uses the machine code program to draw a gray line along the horizon. And line 1060 uses it again to color in the grass.

```
2000 REM ***     CASTLE-MAIN BODY     ***
2010 Y1=10:Y2=16:X1=14:X2=25:C=2:GOSUB 13000
2020 FORI=0TO69:  X=RND(0)*48+56
2030 SYS MC,X,X,RND(0)*56+80,2
2040 NEXT I
```

The subroutine that draws the main body of the castle uses our box drawing routine at line 13000. So before that routine is called line 2010 specifies the sides of the box and gives it a color – **2**, which is red. **NS** – the no-stripe variable – is not set – so the red-brick of castle is striped with orange mortar by the box routine. This doesn't look too realistic so we add random red bricks with lines 2020 through 2040, to break up the pattern and give a mottled effect.

```
3000 REM *** TURRETS LEFT & RIGHT  ***
3010 Y1=6:Y2=11:X1=12:X2=15:C=2:GOSUB13000
3020 X1=24:X2=27: GOSUB13000
3030 FOR I=0 TO 39
3040 X=RND(0)*16+48:SYS MC,X,X,RND(0)*48+48,2
3050 X=RND(0)*16+96:SYS MC,X,X,RND(0)*48+48,2
3060 NEXT I
4000 REM *** `     TURRET TOPS       ***
4010 FORI=1 TO 5: FORJ=11 TO 16
4020 POKE SC+I*40+J,11: NEXTJ,I
4030 FORI=1TO9: FORJ=0TO3
4040 SYS MC,56-I,55+I,8+I*4+J,2: NEXTJ,I
4050 FORI=1TO5: FORJ=23TO28
4060 POKE SC+I*40+J,11: NEXTJ,I
4070 FORI=1TO9: FORJ=0TO3
4080 SYS MC,104-I,103+I,8+I*4+J,2: NEXTJ,I
```

To draw the turrets, the box-drawing routine at line 13000 is
called again. But this time it is called twice, once for each
turret. Again the top, bottom and sides of the box must be
specified. But the second time the box-drawing routine is
called, only new sides for the turret have to be specified.
The top and bottom are at the same level. The boxes are
filled in with color 2, red, again. And again **NS** is not set to 1,
so the orange mortar stripes are added automatically. Lines
3040 and 3050 gives the random speckling of red bricks. To
put the turret tops on, the screen memory in that area is
**POKE**d with 11, which is gray, and lines 4040 and 4080 draw
up a series of gray lines which give the pointed turret roofs.

*SYS MC ACTS LIKE
A GOSUB BUT
INSTEAD OF DOING
A PART OF YOUR
BASIC PROGRAM
IT EXECUTES SOME
MACHINE CODE.*

```
5000 REM ***         MAIN ROOF          ***
5010 FOR I=8 TO 9: FOR J=16 TO 23
5020 POKE SC+I*40+J,11: NEXT J,I
5030 FOR Y=70 TO 79: SYS MC,64,95,Y,2: NEXT Y
```

The main roof is drawn in a similar way. The number
corresponding to the color gray, 11, is **POKE**d into the
screen memory in the area occupied by the roof, and **SYS
MC** calls the machine code routine repeatedly. This draws
up a series of gray lines down the screen which form the
roof. The main roof is square, though, and simpler to draw.
So instead of the two loops which give lines which get longer
as the routine moves on the screen – giving the turrets'
pointed roofs – only one loop is needed. Notice how this way
of drawing a block differs from using the **DRAW BOX**
routine. Both approaches can be used later when you are
writing your own programs.

```
6000 REM ***        BATTLEMENTS        ***
6010 FOR I=0 TO 1: FOR J=0 TO 1
6020 POKE 1441+I*4+J,11: NEXT J,I
6030 FOR Y=80 TO 87
6040 SYS MC,68,75,Y,2: SYS MC,83,91,Y,2
6050 NEXT Y
7000 REM ***    WINDOWS AND DOOR    ***
7010 POKEV+21,240
7020 Y1=14:Y2=16:X1=19:X2=20
7030 C=9:NS=1:GOSUB 13000
7040 RETURN
```

To draw in the battlements, the roof color is simply extended down into the wall in two square areas. So gray is **POKE**d into that area of the screen memory and **SYS MC** is called again twice. The **2** at the end of the string of figures after each **SYS MC** command tells the machine code program to get its color from the screen memory, rather than the color memory or anywhere else.

The windows have already been defined as sprites in the initialization routine. So to draw in the windows we only have to turn the window sprites on with the instruction in line 7010. The door, though, is drawn up using the box drawing routine at line 13000. The sides of the door are set in line 7020 and line 7030 sets the color to **9**, brown. Since we don't want orange stripes on the door, the value of **NS** is set back to **1**, then the box-drawing routine is called.



THE COMMODORE
WINDOW SPRITE
IN DETAIL.

When sprites are defined, each is given a pointer which tells the sprite where to find its **DATA** in the computer's memory. This **DATA** has to be put into memory separately. Here is the **DATA** for the window sprites. It is **READ** and **POKE**d into memory in the right place by the initialization program. Every number in this **DATA** defines a small part of each window when it appears on the screen. So be careful that you type it in absolutely right or the program will not work properly. It is very easy to make a mistake when you are keying in a long string of numbers. Be sure to double check it.

```
20560 REM   SPRITE DATA:WINDOW
20570 DATA 255,252,0,86,84,0,250,188,0,90,148,0,250
20580 DATA 188,0,90,148,0,250,188,0,90,148,0,250,188,0
20590 DATA 90,148,0,250,188,0,90,148,0,250,188,0,90
20600 DATA 148,0,250,188,0,90,148,0,250,188,0,90,148,0
20610 DATA 250,188,0,90,148,0,255,252,0,000
```
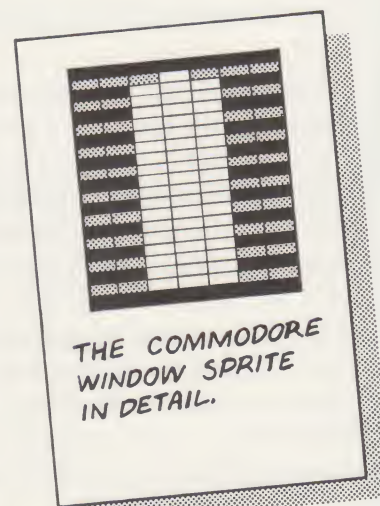
```
8000 REM ***            TREES              ***
8010 FOR TR=0 TO 9
8020 TY=INT(RND(0)*46)+120: TX=INT(RND(0)*144)
8030 IF (TX>40 AND TX<104) AND TY<136 THEN 8020
8040 FOR I=20 TO 0 STEP-1: Y=TY+I
8050 FOR J=0 TO 15: X=TX+J
8060 MO=TR(I,J): IFMO<>2 THEN 8080
8070 SYS MC,X,X,Y,MO
8080 NEXT J,I,TR
8090 RETURN
```

Now that the scene has been set and the castle is in place we need to add the spooky forest. One point about the forest is that it is never the same twice when you **RUN** the program. The ten trees are drawn up in random positions which are fixed in line 8020. But we don't want the trees appearing in the sky or in the castle. So they are confined to the bottom and roughly the middle area of the grass by line 8030. If a **TX** or a **TY** position come up that are off limits, line 8030 sends the computer back to line 8020 to roll the **RND** dice again.

The area that each of the trees occupies is stepped across a point at a time by **I** and **J**. **I** moves up and down the tree, while **J** moves across it from side to side. **I** and **J**, of course, start working from the screen positions **TX** and **TY** worked out before. At each new position and a new element of the tree array **TR** is read into **MO**. If its not equal to **2**, the computer skips the draw instruction and moves on to the **NEXT** point. But if it is **2**, **SYS MC** draws in the tree trunk in brown, which is the color that has been **POKE**d into the screen memory of this area of the screen.

## RUN
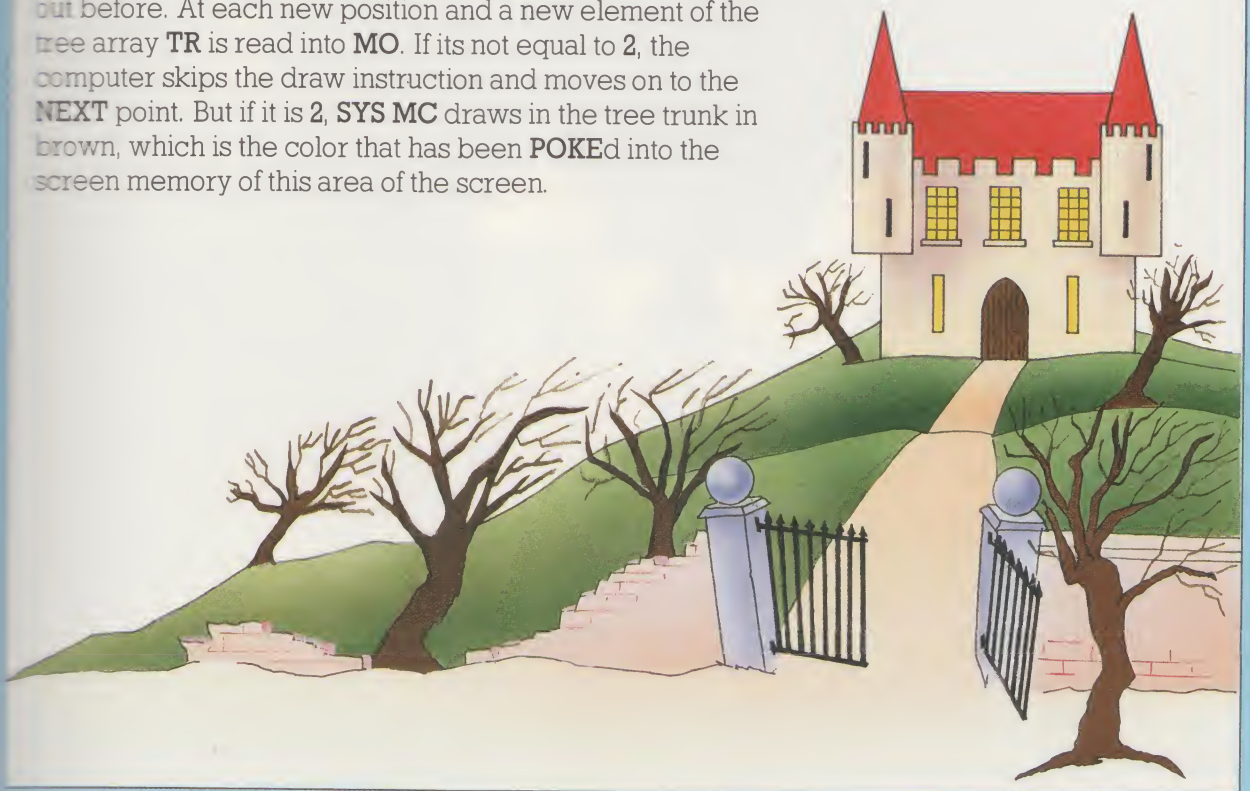
**Testing your program**

You have now completed the part of the program that draws up the scenery, the castle and the spooky forest, so it will be as well to test them and check that they are working properly before the after-dark action starts. To do that, you'll have to add the following test lines though. Otherwise the main routine will call subroutines that have not yet been typed in, and you will receive an error message.
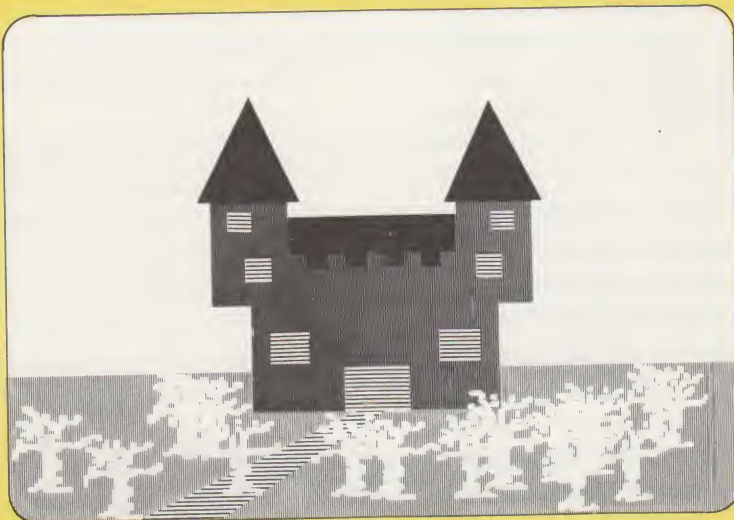


### APPLE

TYPE 125 **GOTO** 125 AND **RUN** THE PROGRAM. PRESS THE **CONTROL** AND **RESET** KEYS WHEN YOU ARE SATISFIED THAT THE PROGRAM IS WORKING CORRECTLY. DELETE LINE 125 BEFORE CONTINUING WITH THE NEXT SECTION.
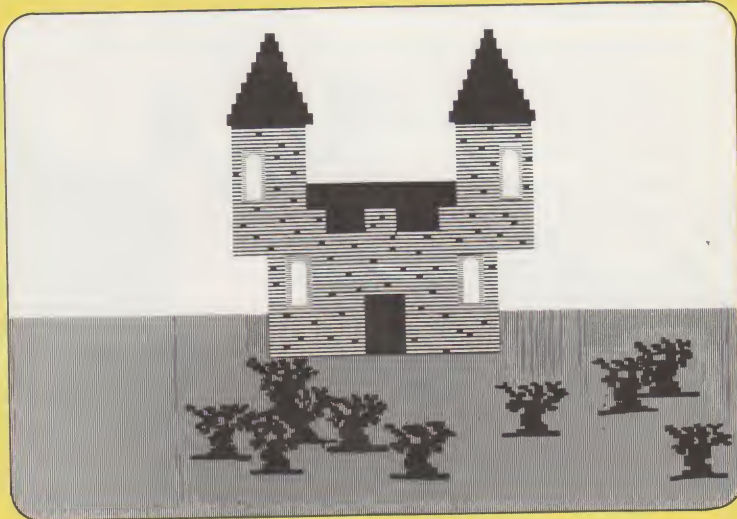


### COMMODORE 64

TYPE 45 **GOTO** 45 AND **RUN** THE PROGRAM. PRESS THE **RUN STOP** AND **RESTORE** KEYS WHEN YOU ARE SATISFIED THAT THE PROGRAM IS WORKING CORRECTLY. DELETE LINE 45 BEFORE CONTINUING WITH THE NEXT SECTION.

# MIDNIGHT

This section of the program turns day into night. The bat, the witch and the burning flames appear. They are animated in different ways, depending on which computer you are using. But once you have learnt these techniques, you can use them in your own programs to create a variety of effects.
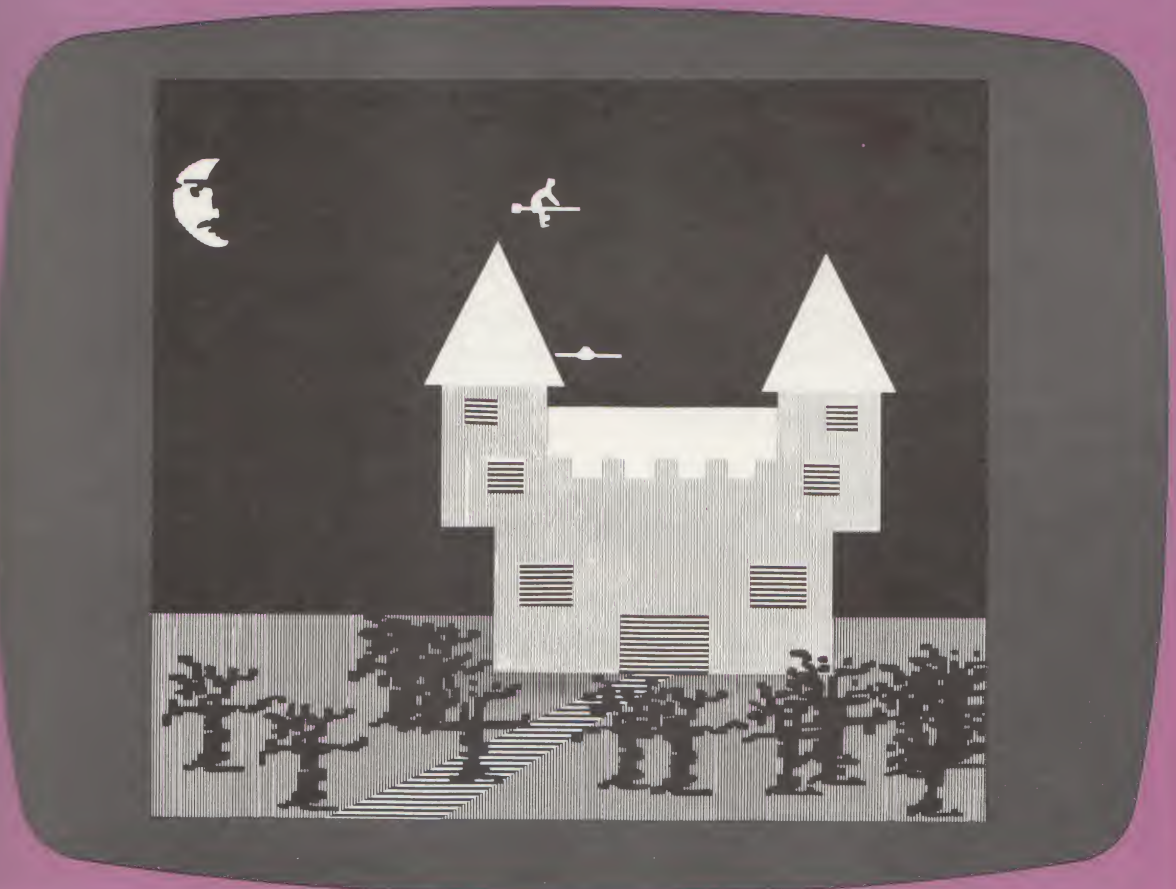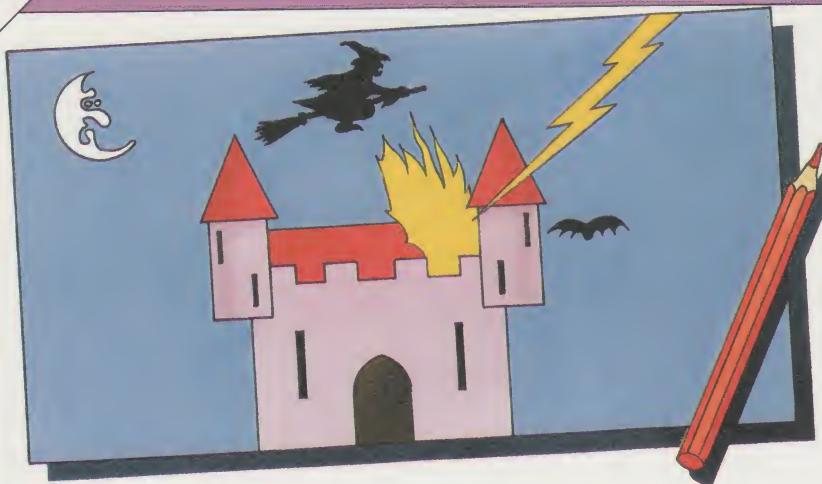
```
5200    REM PLACE MACHINE CODE
5210    READ L,C:T = 0:MACHINECODE = P
5220    FOR I = 1 TO L: READ X:T = T + X
5230    POKE P,X:P = P + 1
5240    NEXT : IF T = C THEN  RETURN
5250    PRINT : PRINT "CHECK LINES  5300 -> 5360": END
5300    DATA 57,5955,169, 32,133,  9,169, 64,133
5310    DATA  11,169,  0,133,  8,133, 10,168,169
5320    DATA  85,133, 12, 74,133, 13,177,  8, 72
5330    DATA  41,128,240,  9,152, 41,  1,170,104
5340    DATA  85, 12,208,  3,104, 73,127,145, 10
5350    DATA 200,208,232,230, 11,230,  9,165,  9
5360    DATA 201, 64,208,222, 96
```

The machine code routine that switches day into night is contained in the **DATA** in lines 5300 through to 5360. Lines 5220 through 5240 **READ** the **DATA** and **POKE** it into memory. When you type in machine code you must get the figures exactly right. The routine adds up all the figures and checks them against a check sum – if you make a mistake, the message in line 5250 will be displayed.

When night falls the routine at line 5000 is called. It immediately calls the machine code which copies the daytime scene onto graphics screen 2.

```
5000    REM CAUSE NIGHTFALL
5010    CALL MACHINECODE: REM CREATE NIGHT ON SCRN 2
5020    POKE GPAGE,64: REM DRAW ON 2ND SCREEN
5030    HCOLOR= 3: DRAW 2 AT 15,7: REM MOON
5040    FOR X = 1 TO 2000: NEXT :X =  PEEK (P2GE)
5050    POKE GPAGE,32: GOSUB 5100: REM LIGHTNING ON S1
5060    POKE GPAGE,64: RETURN : REM NEXT PLOTS ON S2
```

```
4200    REM MOON
4210    DATA P5L3R  D6LDL  6RD7L  DL7RD  8LDRD  LM3RP  3RM3R
4220    DATA DP2LM  5LP3L  D4RD4  LD10R  UU3LR  DRRDD  10LD1
4230    DATA ORDRR  12LRD  11RLD  10LRD  8RMD4  LP4LR  D3RM2
4240    DATA RP2RL  D5LRD  6RDR6  LRRD7  RLD4L  E
```

POKEing **GPAGE** with **64** in line 5020 allows you to draw on the second screen. Remember that the first graphics screen is still the one that is appearing on the TV screen. The first thing to be drawn on screen two is the moon. The **DATA** for the shape of the moon is read in from lines 4210 through 4240. Like the rest of the shape **DATA**, it ends with an **E**. When the shape-table routine **READ**s an **E**, it knows that it has reached the end of the shape. This is shape two, so the **DRAW 2** instructions in line 5030 draws the moon up. It is drawn in color **3**, given by the **HCOLOR** command and the moon is drawn at coordinates 15, 7. Next there's a 2,000 time period pause. Then, by **PEEK**ing **P2GE**, the second graphics screen appears on the TV. Line 5050 **POKE**s **GPAGE** with **32** which allows you to draw on the first screen again. The computer then goes off to the subroutine at line 5100 to draw in the lightning. When it has done that **POKE**ing **GPAGE** with **64** lets you draw on screen 2 again.

The lightning is drawn on graphics screen 1 by lines 5100 through to 5150 while the other screen is displayed on the TV. The lightning is drawn in color **7**, white, and is made of a long narrow triangle which thins toward the bottom. Every 23 lines down the screen this triangle is given a quick shift to the left to give it the proper jagged look of lightning.

```
5100    HCOLOR= 7:DX = 0:C = 0: REM DRAW LIGHTNING
5110    FOR Y = 1 TO 100:C = C + 1
5120    LET D1 = Y * .96:D2 = Y * 1.25
5130    IF C = 23 THEN DX = DX + 29 + D1 - D2:C = 0
5140    HPLOT 250 - D1 + DX,Y TO 279 - D2 + DX,Y
5150    NEXT : RETURN
```

To flash the lightning, the screen is changed to screen 1 by **PEEK**ing **P1GE**. This displays the day screen with the lightning drawn on it on the TV.

```
2400    REM FLASH LIGHTNING
2410    LET X =  PEEK (P1GE)
2420    FOR X = 1 TO 60
2430    NEXT :X =  PEEK (P2GE): RETURN
```

```
2200    REM BAT ROUTINES
2210    HCOLOR= 3:BN = 0:BA = 3:BR = 2:BX = 70
2220    GOSUB 2260: XDRAW 3 + BN AT BX,BY: RETURN
2230    HCOLOR= 3:X = BX:Y = BY: GOSUB 2260
2240    XDRAW 3 + BN AT X,Y:BN = 1 - BN
2250    XDRAW 3 + BN AT BX,BY: RETURN
2260    LET BX = BX + 2 *  COS (BA) +  RND (1)
2270    LET BY = BR *  SIN (2 * BA) + 65
2280    LET BA = BA + .1:BR = 2 *  LOG (BA): RETURN
2300    HCOLOR= 3: REM REMOVE BAT
2310    LET X = BX:Y = BY:BY = BY - 1: GOSUB 2240
2320    FOR X = 1 TO 50: NEXT : IF BY > 2 THEN 2310
2330    GOTO 2250
```

To make the bat appear to flap its wings there are two bat shapes in the shape table: one with the bat's wings up and one with the bat's wings down. The shapes are switched by the **BN=1−BN** sum. The **XDRAW** command is an "exclusive" draw. This means that drawing a new bat shape will blank out the old one. The path that the bat takes when it flies is a spiral, fixed by the instructions in lines 2260 through 2280. The subroutine that removes the bat calls the same drawing routine but its path is fixed by lines 2310 through 2330. When this routine is called, these lines make the bat fly off the top of the screen.

```
4300    REM BATS 1 & 2
4310    DATA MUURP RRDRD 3RURU RDRD3 LD3RU 3RURU RRE
4410    DATA P6RUR URDRD 3LD3R U6RE
```

The **DATA** for the two bat shapes is given in lines 4310 and 4410. Note that there are two "**Es**" in the **DATA**, so that the computer knows that there are two shapes. Draw the two bats on graph paper, using the **DATA**, in the way explained on page 30. Then along comes the witch.

```
2100    REM WITCH ROUTINES
2110    HCOLOR= 7:WX = 30:WY = 10
2120    XDRAW 5 AT WX,WY: RETURN
2150    HCOLOR= 7: REM MOVE WITCH
2160    LET X = WX: IF WX < 0 THEN  RETURN
2170    IF WX = 260 THEN WX =  - 1: GOTO 2190
2180    LET WX = WX + 1: XDRAW 5 AT WX,WY
2190    XDRAW 5 AT X,WY: RETURN
```

```
4500    REM WITCH
4510    DATA PRDLD LLMLD P3RD4 LDP2R M2RPR DRDRM 4RDP1
4520    DATA 8LD3R MUULP 3LM7R UP2RD 2LM2D RP2RD RDLDD
4530    DATA RE
```

In the witch routine, the **XDRAW** instruction is used to draw
the witch over the background, but the background is
returned intact when she has passed by. When the witch
moves, two **XDRAWS** are used one after the other. The first
draws up the new witch, the second blanks out what was left
of the old one. Line 2170 detects when the witch has reached
the edge of the screen, skips the new drawing instruction
and goes straight to the blanking out line.

```
2500    LET H = 2:W = 2: REM ANIMATE FIRE
2510    FOR I = 1 TO 400
2520    IF H < 70 THEN H = H + 1
2530    IF W < 118 THEN W = W + 1
2540    LET X =  RND (1) * W:Y =  RND (1) * H
2550    IF I / (X + Y) < 2 *  RND (1)  THEN 2540
2560    LET X = OX + 132 - X
2570    HCOLOR= 4 + 2 *  RND (1):Y = 80 + OY - Y
2580    DRAW 3 *  RND (1) + 6 AT X,Y
2590    NEXT
2600    REM CASTLE CRUMBLES
2610    LET X(1) = 60:Y(1) = 0:C(0) = 0
2620    LET X(2) = 200:Y(2) = 110:C(1) = 0
2630    GOSUB 6100: RETURN
```

There are three flame shapes and these are called up
randomly, in random colors, in random positions that
gradually cover the castle. The castle is made to crumble
simply by printing a block of black over it using the
subroutine at line 6100.

```
4600    REM 3 FLAMES
4610    DATA P3LU3 RU3LU 2RU2L URURU LURUE
4710    DATA P3RU3 LU3RU 2LU2R ULULU RULUE
4810    DATA M3UPU 2L4RU 4LRUR RULE
```

```
4900    REM SKULL
4910    DATA P17LD  2LDLD  LDLDL  2DL4D  L7DR4  DR3DR  DRDRD
4920    DATA RDR3D  19R3U  RURUR  URUR3  UR4UR  7UL4U  L2ULU
4930    DATA LULUL  U2LUM  13DPL  U4LDL  6DRD5  R7UM1  1LPLU
4940    DATA 4LDL7  D5RUR  6UM10  DP4D5  R4ULU  3LDLM  6L14D
4950    DATA P2DR4  D15R4  UR2U1  7LE
```

The skull in the sky is enlarged by setting the **SCALE** to **2**.
And it is made to shimmer eerily in the sky by altering the
position it is printed in slightly. The **X** position is shifted
randomly by 3 high-resolution screen positions and the **Y**
position is shifted randomly by 4. Line 2730 rolls the **RND**
dice for these shifts. The skull, which is shape **9**, is then
drawn by **XDRAW**. This also blanks out the last skull that
appeared on the screen. The skull is printed up – and
blanked out – 2,000 times before the computer **RETURN**s
from this subroutine and the whole program **END**s. But, until
then, the effect is very eerie indeed.

```
2700    REM SKULL IN SKY
2710    HCOLOR= 7: SCALE= 2
2720    FOR I = 1 TO 2000
2730    LET X =   RND (1)  * 3:Y =   RND (1)  * 4
2750    XDRAW 9 AT 148 + X,10 + Y
2760    NEXT : RETURN
```

All that is needed now to make the whole program work is
shape-table **DATA** for the skull itself. This is typed in the
code which consists of print and move commands and
directional instructions as before. See if you can trace out the
shape on a piece of graph paper. This will give you a clear
outline of the skull shape, without the shimmer that is given
by shifting its position when it is printed successively on the
screen. And like the rest of the shape **DATA**, the skull's
**DATA** ends with an **E** that tells the part of the initialization
routine which is filling the shape table that this is the end of
the last shape in this program. The machine code routine on
pages 8 and 9 can be used in other programs that you write.
It makes it much easier to create character shapes on the
Apple IIe.
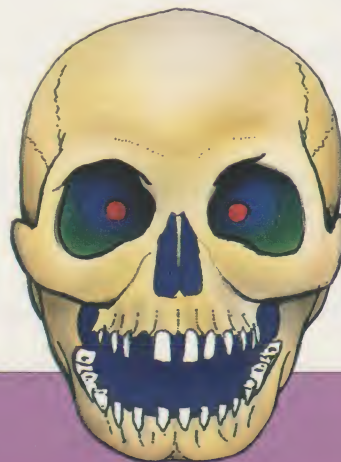
```
9000 REM ***           NIGHT TIME          ***
9010 POKE 53281,0
9020 FOR I=0 TO 8: FOR J=0 TO 3
9030 POKE SC+I*40+J,7
9040 NEXT J,I
9050 FOR I=0 TO 16: Y=5+I
9060 FOR J=0 TO 7: X=5+J
9070 MO=MO(I,J): IF MO>3 THEN 9090
9080 SYS MC,X,X,Y,MO
9090 NEXT J,I
9100 RETURN
```

Making darkness fall is easy. The **POKE** in line 9010 changes the background color – which was the sky's light blue – to black. But then the moon has to appear. Lines 9020 through 9040 **POKE** an area of the screen memory with **7**, the code for the color yellow. Lines 9050 through 9090 draw in the shape of the moon there using the machine code call **SYS MC**. The **DATA** for the moon is read out of the array **MO**.

That same **DATA** is **READ** into the array **MO** by the initialization routine. But at that time we did not give it any **DATA**. Here it is now. You'll notice again that the shape of the moon is given by the 2s. In the night routine, line 9070 skips the machine code call if it hits a **4** and nothing is printed on the screen. The points making up the moon are only printed on the screen when a **2** is found. Each **2** is also used in the **SYS MC** in line 9080 to tell the machine code to take its color from the screen memory.
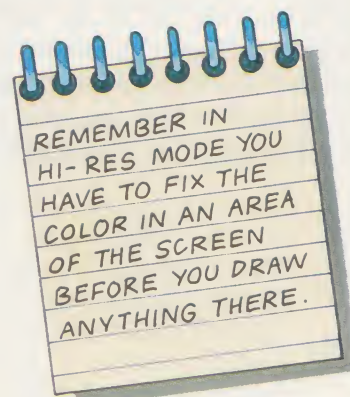
> REMEMBER IN HI-RES MODE YOU HAVE TO FIX THE COLOR IN AN AREA OF THE SCREEN BEFORE YOU DRAW ANYTHING THERE.

```
20220 REM  DATA FOR MOON
20230 DATA4,4,4,4,4,2,2,2
20240 DATA4,4,4,2,2,2,4,4
20250 DATA4,4,4,2,2,4,4,4
20260 DATA4,4,2,2,4,4,4,4
20270 DATA4,2,2,2,4,4,4,4
20280 DATA4,2,2,2,4,4,4,4
20290 DATA4,2,2,4,4,4,4,4
20300 DATA4,2,2,4,2,4,2,4
20310 DATA2,2,2,4,4,4,4,4
20320 DATA2,2,2,4,4,4,4,4
20330 DATA2,2,2,2,2,2,4,4
20340 DATA2,2,2,4,4,2,4,4
20350 DATA2,2,2,4,4,4,4,4
20360 DATA4,2,2,4,2,4,4,4
20370 DATA4,4,2,2,2,2,4,2
20380 DATA4,4,2,2,2,2,2,4
20390 DATA4,4,4,2,2,2,4,4
```

```
10040 REM ***          MOVE BAT         ***
10050 FOR D=0 TO 30: NEXT
10060 T=1-T: POKE 2041,35+T
10070 IF X<1 OR X>69 THEN DX=-DX
10080 IF Y<49 OR Y>99 THEN DY=-DY
10090 IF RS=255 THEN DX=1: DY=0
10100 X=PEEK(V+2)+DX: Y=PEEK(V+3)+DY
10110 POKEV+2,X: POKEV+3,Y
10120 RETURN
```

The subroutine that moves the bat starts with a 30 time period delay. The little sum **T**=**1**−**T** in line 10060 flips the number of the **DATA** block **POKE**d into the bat's sprite pointer between **35** and **36**. Lines 10070 and 10080 change the bat's direction when it has reached its limits. Line 10090 sends the bat off screen when the witch comes by and the rest of the routine fixes the bat's sprite position.

```
20790 REM   BAT - WINGS UP
20800 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,28,0
20810 DATA 56,119,0,238,129,219,129,0,60,0,0,24,0,0,36
20820 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20830 DATA 0,0,0,0,0,0,000
20840 REM   BAT - WINGS DOWN
20850 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20860 DATA 0,0,129,219,129,119,60,238,28,24,53,0,36,0
20870 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20880 DATA 0,0,0,0,0,000
```

The bat needs two lots of **DATA** – one to draw it with its wings up and one with its wings down. These are read into blocks **35** and **36** by the initialization routine. So each time the bat routine is called, the other picture is printed up. Alternating between the two images in this way makes the bat appear to flap its wings as it flies.

The witch flies on her broomstick though. Since she has no wings, she only needs one block of **DATA** for the one image.

```
20730 REM WITCH
20740 DATA 0,0,0,0,60,0,0,14
20750 DATA 0,0,15,0,0,15,192,0,23,0,0,7,128,2,15,0,1
20760 DATA 255,0,3,255,128,4,249,128,0,60,192,160,126
20770 DATA 224,84,126,112,171,255,255,84,191,0
20780 DATA 128,7,0,0,30,0,0,60,0,0,112,0,0,112,0,000
```

```
11000 REM ***            BURN             ***
11010 P(0)=116:P(1)=107:P(2)=228:P(3)=107
11020 P(4)=138:P(5)=147:P(6)=206:P(7)=147
11030 FOR I=0 TO 7: POKE V+I,P(I): NEXT
11040 POKE V+16,0: POKE V+28,255
11050 FOR I=0 TO 3:POKE2040+I,33:POKEV+39+I,2:NEXT
11060 POKE V+21,255:T=0
11070 FOR BU=1 TO 200: T=1-T
11080 POKE V+37,0+7*T: POKE V+38,7-7*T
11090 IF BU=100 THEN GOSUB 11140
11100 IF BU=50 THEN GOSUB 11160
11110 FOR D=0 TO 40:NEXT
11120 NEXT BU:POKE V+21,0
11130 FOR Y=0 TO 117:SYS MC,40,120,Y,0:NEXT:RETURN
11140 FOR I=0 TO 3: POKE V+2*I,P(2*I)-12
11150 POKE V+29,15: NEXTI: RETURN
11160 FOR I=0 TO3:POKE V+1+2*I,P(2*I+1)-20
11170 POKE V+23,15: NEXTI: RETURN
```

Lines 11010 and 11020 give the coordinates of the flames.
Line 11040 sets the sprites to high-resolution mode, and line
11050 sets their color to **2**, or red. Once the flame sprites are
turned on, the burn time, **BU**, is set to 200. Line 11080 flips the
sprite's high-resolution color between yellow and black to
make them flicker. When **BU** reaches 50, and again at 100,
the program branches to subroutines which make the sprites
expand. But when that happens the sprites have to be
repositioned. Line 11030 finally burns the place down.

```
20650 REM   FLAME
20660 DATA  0,192,0,0,4,0,1,0,0,1,204,0,5,196,0,5,196,0
20670 DATA  5,76,0,7,100,0,7,164,0,7,225,0,1,225,0,1
20680 DATA  233,0,192,232,112,48,104,112,60,232,112
20690 DATA  25,229,64,58,109,192,14,229,192,14,105
20700 DATA  192,7,186,180,1,174,180,255
```

All the flames are the same shape, so they only need one lot
of **DATA**. This is **POKE**d into memory by the initialization
program. They don't all look the same on the screen though,
because of the use of constantly changing colors.

```
10000 REM ***         LIGHTNING         ***
10010 FOR LI=1 TO 0 STEP-1
10020 POKE53281,LI: FOR D=0 TO 50: NEXTD,LI
10030 RETURN
```

All the lightning subroutine does is to make the lightning
flash. The lightning appears when 53281 is **POKE**d with 1.
After a pause caused by the machine counting to 50, the
lightning disappears when it is changed back to black again.

```
12000 REM ***         SKULL         ***
12010 FOR I=1 TO 12: FOR J=15 TO 24
12020 POKE SC+I*40+J,RND(0)*255: NEXTJ,I
12030 FOR I=14 TO 0 STEP -1: FOR J=0 TO 9
12040 MO=RND(0)*2+1: IF SK(I,J)=4 THEN 12080
12050 X=60+J*4: Y=40+I*4
12060 SYS MC,X,X+2,Y,MO
12070 SYS MC,X,X+2,Y+1,MO
12080 NEXT J,I
12090 RETURN
```

To create the eerie skull, lines 12010 and 12020 **POKE**
random color values into the screen memory in the area that
is going to be occumped by the skull. The **DATA** for the skull
is read out of the array **SK**. If the element of the array for that
particular area is **4**, the drawing lines are skipped. Otherwise,
random numbers between **1** and **3** direct the machine code
to pick its colors from different areas of memory.
The 0s in the skull data draw out the shape of the skull and
the 4s are left as the background.

```
20400 REM  DATA FOR SKULL
20410 DATA 4,4,4,0,0,0,0,4,4,4
20420 DATA 4,0,0,0,0,0,0,0,0,4
20430 DATA 4,0,0,0,0,0,0,0,0,4
20440 DATA 0,0,0,0,0,0,0,0,0,0
20450 DATA 0,0,4,0,0,0,0,4,0,0
20460 DATA 0,0,4,4,0,0,4,4,0,0
20470 DATA 0,0,4,4,0,0,4,4,0,0
20480 DATA 0,0,0,0,0,0,0,0,0,0
20490 DATA 4,0,0,0,4,0,0,0,0,4
20500 DATA 4,0,0,0,4,4,0,0,0,4
20510 DATA 4,0,0,0,0,0,0,0,0,4
20520 DATA 4,4,4,0,0,0,0,4,4,4
20530 DATA 4,4,0,0,0,0,0,0,4,4
20540 DATA 4,4,0,4,4,4,4,0,4,4
20550 DATA 4,4,0,0,0,0,0,0,4,4
```
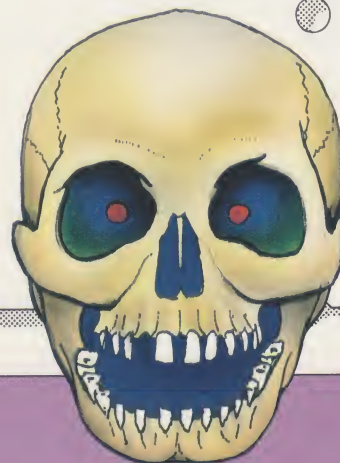
## Program listings

What follows is the complete program for both computers.
List the program and check it against the full listing here.
Even the smallest mistake can cause problems. Take special
care to check the **DATA** lines. A comma out of place here
can stop the whole program working. When you have a full,
working program, **SAVE** it on tape or disk, using the
commands in your user's manual.

**APPLE IIe**

**COMMODORE 64**

```
1 REM HAUNTED HOUSE
5 DEF  FN MOD(X) = 256 * (X / 256 - INT (X / 256))
100 GOSUB 1000: REM INITIALIZATIONS
110 GOSUB 6000: REM BACKGROUND & CASTLE
120 GOSUB 6500: REM TREES
130 GOSUB 5000: REM NIGHTFALL
140 GOSUB 2400: REM LIGHTNING
150 FOR TI = 1 TO 250
160 IF TI = 10 THEN  GOSUB 2100: REM START WITCH
170 IF TI > 10 THEN  GOSUB 2150: REM MOVE WITCH
180 IF TI = 30 THEN  GOSUB 2200: REM START BAT
190 IF TI > 30 THEN  GOSUB 2250: REM MOVE BAT
200 IF  RND (1) < .07 THEN  GOSUB 2400: REM LIGHTNING
210 NEXT TI: GOSUB 2300: REM REMOVE BAT
220 GOSUB 2400: GOSUB 2400: REM STORM
230 GOSUB 2500: REM FIRE
240 FOR X = 1 TO 1000: NEXT : GOSUB 2700: REM SKULL
250 TEXT : END
1000 REM INITIALIZE PEEK AND POKE ADDRESSES
1010 LET P1GE = 49236:P2GE = 49237
1020 LET FULLSCREEN = 49234:GPAGE = 230
1030 LET CLICK = 49200:P = 24576
1040 GOSUB 3000: REM SET UP SHAPE TABLE
1050 GOSUB 5200: REM PUT MACHINE CODE IN
1060 HGR :X =  PEEK (FULLSCREEN)
1070 SCALE= 1: ROT= 0: RETURN
2100 REM WITCH ROUTINES
2110 HCOLOR= 7:WX = 30:WY = 10
2120 XDRAW 5 AT WX,WY: RETURN
2150 HCOLOR= 7: REM MOVE WITCH
2160 LET X = WX: IF WX < 0 THEN  RETURN
2170 IF WX = 260 THEN WX = - 1: GOTO 2190
2180 LET WX = WX + 1: XDRAW 5 AT WX,WY
2190 XDRAW 5 AT X,WY: RETURN
2200 REM BAT ROUTINES
2210 HCOLOR= 3:BN = 0:BA = 3:BR = 2:BX = 70
2220 GOSUB 2260: XDRAW 3 + BN AT BX,BY: RETURN
2240 HCOLOR= 3:X = BX:Y = BY: GOSUB 2260
2240 XDRAW 3 + BN AT X,Y:BN = 1 - BN
2250 XDRAW 3 + BN AT BX,BY: RETURN
2260 LET BX = BX + 2 *  COS (BA) +  RND (1)
2270 LET BY = BR *  SIN (2 * BA) + 65
2280 LET BA = BA + .1:BR = 2 *  COS (BA): RETURN
2300 HCOLOR= 3: REM REMOVE BAT
2310 LET X = BX:Y = BY:BY = BY - 1: GOSUB 2260
2320 FOR X = 1 TO 50: NEXT : IF BY < 2 THEN 2310
2330 GOTO 2250
2400 REM FLASH LIGHTNING
2410 LET X =  PEEK (P1GE)
2420 FOR X = 1 TO 60
2430 NEXT :X =  PEEK (P2GE): RETURN
2500 LET H = 2:W = 2: REM ANIMATE FIRE
2510 FOR I = 1 TO 400
2520 IF H < 70 .THEN H = H + 1
2530 IF W < 118 THEN W = W + 1
2540 LET X =  RND (1) * W:Y =  RND (1) * H
2550 IF I / (X + Y) < 2 *  RND (1) THEN 2540
2560 LET X = OX + 132 - X
2570 HCOLOR= 4 + 2 *  RND (1):Y = 80 + OY - Y
2580 DRAW 3 *  RND (1) + 6 AT X,Y
2590 NEXT
2600 REM CASTLE CRUMBLES
2610 LET X(1) = 60:Y(1) = 0:C(0) = 0
2620 LET X(2) = 200:Y(2) = 110:C(1) = 0
2640 GOSUB 6100: RETURN
2700 REM SKULL IN SKY
2710 HCOLOR= 7: SCALE= 2
2720 FOR I = 1 TO 2000
2730 LET X =  RND (1) * 3:Y =  RND (1) * 4
2750 XDRAW 9 AT 148 + X,10 + Y
2760 NEXT : RETURN
3000 READ SN: REM SHAPE TABLE SET UP
3010 POKE P,SN: POKE P + 1,0:ST = P + 2
3020 POKE 232, FN MOD(P): POKE 233, INT (P / 256)
3030 LET P = ST + 2 * SN
3040 FOR SS = 0 TO SN :I = P - ST + 2
3050 LET A = ST + 2 * SS: POKE A, FN MOD(T)
3060 POKE A + 1, INT (T / 256):D$ = "":R = 0
3070 POKE P,25:P = P + 1: GOSUB 3150: NEXT : RETURN
3100 LET X = X + B: FOR I = 1 TO R
```

```
5 REM ***    STARTER PROGRAM      ***
10 FOR I=0 TO 214: READ D: CS=CS+D
20 POKE 49152+I,D: NEXT I
30 IF CS<>29713 THEN PRINT "DATA ERROR!":END
40 POKE642,64:SYS64766
100 DATA0,32,253,174,32,158,183,134,253,32,253,174
110 DATA32,158,183,138,72,32,253,174,32,158,183,138
120 DATA41,7,133,254,138,74,74,74,133,251,10,10
130 DATA101,251,160,6,162,0,134,252,42,38,252,136
140 DATA208,250,101,254,133,251,165,253,41,252,10,144
150 DATA3,230,252,24,101,251,133,251,165,252,105,32
160 DATA133,252,32,253,174,32,158,183,138,141,0,192
170 DATA160,3,10,10,13,0,192,136,208,248,141,0
180 DATA192,165,253,74,74,133,255,104,133,254,74,74
190 DATA56,229,255,240,52,133,255,32,178,192,32,149
200 DATA192,169,8,74,101,133,251,133,251,165,252,105,0
210 DATA133,252,138,255,240,8,173,0,192,145,251,76
220 DATA121,192,32,197,192,72,73,255,49,251,145,251
230 DATA104,45,0,192,17,251,145,251,96,32,178,192
240 DATA133,255,32,197,192,37,255,76,149,192,169,255
250 DATA72,165,253,41,3,240,8,173,104,74,74,202
260 DATA208,251,96,104,96,169,255,72,169,252,5,254
270 DATA170,104,232,208,1,96,10,10,76,206,192

5 REM ***  ==HAUNTED HOUSE==   ***
10 GOSUB 14000:REM INITIALIZE
20 GOSUB 1000 :REM BACKGROUND
30 GOSUB 2000 :REM CASTLE
40 GOSUB 8000 :REM TREES
50 GOSUB 9000 :REM NIGHT
60 POKE V+21,247:Dx=1:Dy=1
70 FOR WX=10 TO 246 STEP 2:LI=RND(0)*20
80 IF LI<19 THEN GOSUB 10000:REM LIGHTNING
90 IF WX<255 THEN RS=C55:POKE V+16,3
100 POKE V+0,WX-RS:REM WITCH
110 GOSUB 10040:REM BAT
120 NEXT WX: POKE V+21,240
130 GOSUB 11000:REM BURN
140 GOSUB 12000:REM SKULL
150 END
1000 REM ***      BACKGROUND        ***
1010 FOR I=0 TO 39: POKE 53896+I,11: NEXT
1020 FOR I=10 TO 24: FOR J=0 TO 39
1030 POKE CO+I*40+J,5: POKE SC+I*40+J,121
1040 NEXT J,I
1050 SYS MC,0,15V,111,3
1060 FOR Y=10 TO 199: SYS MC,0,159,Y,3
1070 NEXT Y: RETURN
2000 REM ***   CASTLE-MAIN BODY    ***
2010 Y1=10:Y2=16:X1=14:X2=25:C=.:GOSUB 13000
2020 FORI=0 TO 67: X=RND(0)*48+56
2030 SYS MC,X,X,RND(0)*56+80,3
2040 NEXT I
3000 REM *** TURRETS LEFT & RIGHT ***
3010 X1=6:X2=11:X1=12:X2=15:C4=:GOSUB 13000
3020 X1=24:X2=27: GOSUB 13000
3030 FOR I=0 TO 29
3040 Y=RND(0)*16+48:SYS MC,X,X,RND(0)*48+48,2
3050 X=RND(0)*16+96:SYS MC,X,X,RND(0)*48+48,3
3060 NEXT I
4000 REM ***     TURRET TOPS       ***
4010 FORI=1 TO 5: FORJ=11 TO 16
4020 POKE SC+I*40+J,11: NEXTJ,I
4030 FORI=110 9: FORJ=0 TO J
4040 SYS MC,56-I*55+I,8+I*4+J,2: NEXTJ,I
4050 FORI=1 TO5: POKE SC+I*40+J,11: NEXTJ,I
4060 POKE SC+I*40+J,11: NEXTJ,I
4070 FORI=1 TO 9: FORJ=0 TO3
4080 SYS MC,104-I,103+I,8+I*4+J,2: NEXTJ,I
5000 REM ***     MAIN ROOF         ***
5010 FOR I=8 TO 9: FOR J=16 TO 23
5020 POKE SC+I*40+J,11: NEXT J,I
5030 FOR Y=70 TO 79: SYS MC,64,95,Y,2: NEXT Y
6000 REM ***    BATTLEMENTS        ***
6010 FOR I=0 TO 1: FOR J=0 TO 1
6020 POKE 1441+I*4+J,11: NEXT J,I
6030 FOR Y=80 TO 87
6040 SYS MC,68,75,Y,2: SYS MC,83,91,Y,2
```

```
3110 IF PEEK (P - 1)   8 OR X   8 THEN 3130
3120 POKE P - 1,8 * X + PEEK (P - 1): GOTO 3140
3130 POKE P,X:P = P + 1
3140 NEXT :R = 0
3150 IF D$ = "" THEN READ D$: PRINT ".";
3160 LET A$ = LEFT$ (D$,1)
3170 LET D$ = MID$ (D$,2, LEN (D$) - 1)
3180 IF A$ = "R" THEN X = 1: GOTO 3100
3190 IF A$ = "D" THEN X = 2: GOTO 3100
3200 IF A$ = "L" THEN X = 3: GOTO 3100
3210 IF A$ = "U" THEN X = 128: GOTO 3100
3220 IF A$ = "M" THEN R = 0: GOTO 3150
3230 IF A$ = "P" THEN R = 4: GOTO 3150
3240 IF A$ = " " THEN 3150
3250 IF A$ < "0" OR A$ > "9" THEN 3270
3260 LET R = 10 * R + VAL (A$): GOTO 3150
3270 POKE P,X: POKE P + 1,0:P = P + 2: RETURN
4000 REM 9 GRAPHIC ELEMENTS
4010 DATA 9
4100 REM  TREE
4110 DATA P2LUD 9LM3L P2LMU 2RE7R MRU7R 1MU2R P6RU5
4120 DATA L14RM ULP3L MURE5 RMRUP 4LU4R U4LML UP4RU
4130 DATA 4LU4R U4LML UR7RM 5RUP2 LRLLP 14LD3 LDL3D
4140 DATA MR8UP DR2D6 RM3RP 9RMRU P11LU ULM3L P2LMU
4150 DATA RPRU5 LULM3 RP3RU 2LU2L U21UM 4U2RP 2RDRD
4160 DATA RLDRD RLDRD RLDRD RLDLDR MDD3R P4RM2
4170 DATA RP5RU1 5RUR URDDM M1OU5 LP1DL D6LDR 5URUR
4180 DATA M1RP1 3DRU2 D7RU3 DLDRD 2LDRD LDRE
4200 REM MOON
4210 DATA P5L3R D6LDL 6RD7L DI7RD 8LDRD LM3R 3RM3R
4220 DATA DF2LM 5LP3L D4RD4 LD1DR UU3LR DRRDD 1OLDJ
4230 DATA ORDRR 11RRD 11RLD 1O1RD 8RM04 1F4LR D3RM2
4240 DATA RP2RI DSLRD 6RDR6 LRRD7 RLD4L E
4300 REM BAT 1 & 2
4310 DATA MRURI RIDRD SRURI RDRD7 D4RD 7RURU RRE
4410 DATA P2RUR URDRD 3LD7R U6RE
4500 REM WITCH
4510 DATA RMDLD L1NLD L3MD4 LDP2R MDRRM DRDRM 4RDF1
4520 DATA BLD3R MUUL5 3LM7R UP3RD 7LM2D RRDRD RDLDD
4530 DATA RE
4600 REM 3 FLAMES
4610 DATA P3LU5 RU5L 2RP0L DRDRD LURUE
4710 DATA P3RUR LU3RU 2LU2R ULULU RULUE
4810 DATA M3UMU 2L4RU 4LRUR RULE
4900 REM SKULL
4910 DATA P27LD 2LDLD LDLDL 2D14D L7DR4 D43DR DRDRD
4920 DATA RDR3D 1YR3U RURUR URURU DR4UR 2U14U L2UUU
4930 DATA LULUL U2LUM L3DR4 U4LDI 6DRD5 R7UM1 11PLD
4940 DATA 4LDUL D5RU9R 6UM1O DP4D5 R4UUU 3L0LM 6L14D
4950 DATA F2DR4 D15R4 UR5U1 1H
5000 REM CAUSE NIGHTFALL
5010 CALL MACHINECODE: REM CREATE NIGHT ON SCRN =
5020 POKE GPAGE,64: REM DRAW ON 2ND SCREEN
5030 HCOLOR= 3: DRAW 3 AT 15,7: REM MOON
5040 FOR X = 1 TO 2000: NEXT :X =   PEEK (P2GE)
5050 POKE GPAGE,32: GOSUB 5100: REM LIGHTNING ON S1
5060 POKE GPAGE,64: RETURN : REM NEXT PLOTS ON S2
5100 HCOLOR= 3:D = U2:C = 0: REM DRAW LIGHTNING
5110 FOR Y = 0 TO 100:C = 1 + 1
5120 LET D1 = Y * .96:D2 = Y * 1.25
5130 IF C = 23 THEN D1 = D1 + 28 + D1 - D2:C = 0
5140 HPLOT 250 - D1,Y TO 279Y - D2 + D1,Y
5150 NEXT : RETURN
5200 REM PLACE MACHINE CODE
5210 READ L,C:I = 0:MACHINECODE = L
5220 FOR J = 1 TO L: READ X:I = I + X
5230 POKE P,X:P = P + 1
5240 NEXT : IF I < > C THEN RETURN
5250 PRINT : PRINT "CHECK LINES 5300 -> 5360": END
5300 DATA 50,5958,169, 20,141, 5,168, 64,173
5310 DATA 0,160, 0,133, 8,133, 10,168,169
5320 DATA 85,133, 15, 74,151, 15,151, 8, 72
5330 DATA 41,128,240, 9,15, 41, 1,170,104
5340 DATA 8,5,10,208, 1,104, 5,105,145, 10
5350 DATA 200,208, 52,230, 11,250, 9,185, 9
5360 DATA 201, 64,208,55, 96
6000 REM GRAPHIC DATA INTERPRETER
6010 READ R$,X(1),Y(1),X(2),Y(2),X(3),Y(3),C(0),C(1)
6020 IF R$ = "ORIGIN" THEN O = 0:1:0X = X(1):OY = Y(1)
6030 FOR I = 1 TO 3:X(I) = X(I) + OX
6040 LET Y(I) = Y(I) + OY: NEXT I
6050 IF R$ = "BLOCK" THEN GOTO 6100
6060 IF R$ = "TRIANGLE" THEN GOSUB 6200
6070 IF R$ = "STOP" THEN RETURN
6080 GOTO 6010
6100 LET I = - 1: REM FILL BLOCK IN 2 COLOURS
6120 FOR I = Y(1) TO Y(2):I = 1 - I
6130 HCOLOR= C(I): HPLOT X(1),I TO X(2),I
6140 NEXT : RETURN
6200 REM FILL TRIANGLE AREA (X,Y)
6210 LET H = Y(1) - Y(2):H = (X(2) - X(1)) / H
6220 LET I = (X(3) - X(1)) / H:I = 1
6230 FOR I = 0 TO H:I = I + 1: HCOLOR= C(I)
6240 HPLOT X(1) + H * I,Y(1) -
6245 HPLOT 10 * (X(1) + I2 * I,Y(1) - I
6250 NEXT : RETURN
6500 HCOLOR= 3: REM TREES
6510 FOR I = 1 TO 29
6520 LET X =   RND (1) * 250:Y =   RND (1) * 28
6530 DRAW 1 AT X + 25,191 - Y: NEXT : RETURN
7000 REM SKY,GRASS & PATH
7010 DATA ORIGIN,0,0, 0,0,0,0,0
7020 DATA BLOCK,  0, 0,279,175, 0, 0,6,6
7030 DATA BLOCK,  0,136,279,191, 0,  0,1,1
7040 DATA TRIANGLE, 50,191,180,128, 75,191,0,0
7050 REM CASTLE DATA
7060 DATA ORIGIN, 63, 30,0,0,0,0,0,0
```

```
6050 NEXT Y
7000 REM ***    WINDOWS AND DOOR    ***
7010 POKEV+21,240
7020 Y1=14:Y2=16:X1=19:X2=20
7030 C=9:NS=1:GOSUB 13000
7040 RETURN
8000 REM ***        TREES          ***
8010 FOR TR=0 TO 9
8020 TY=INT(RND(0)*46)+120: TX=INT(RND(0)*144)
8030 IF (TX 40 AND TX 104) AND TY 136 THEN 8020
8040 FOR I=20 TO 0 STEP-1: Y=TY+I
8050 FOR J=0 TO 15: X=TX+J
8060 MO=TR(I,J): IF MO 2 THEN 8080
8070 SYS MC,X,X,Y,MO
8080 NEXT J,I,TR
8090 RETURN
9000 REM ***        NIGHT TIME      ***
9010 POKE 53281,9
9020 FOR I=0 TO 8: FOR J=0 TO 5
9030 POKE SC+I*40+J, 7
9040 NEXT J,I
9050 FOR I=0 TO 16: Y=5+I
9060 FOR J=0 TO 7: X=5*4+J
9070 MO=MO(I,J): IF MO 3 THEN 9090
9080 SYS MC,X,X,Y,MO
9090 NEXT J,I
9100 RETURN
10000 REM ***      LIGHTNING        ***
10010 FOR LI=1 TO 0 STEP-1
10020 POKE53281,LI: FOR D=0 TO 50: NEXTD,LI
10030 RETURN
10040 REM ***      MOVE BAT         ***
10050 FOR D=0 TO 30: NEXT
10060 T=1-T: POKE 2041,35+T
10070 IF X 1 OR X 69 THEN DX=-DX
10080 IF Y 49 OR Y 99 THEN DY=-DY
10090 IF RS=255 THEN DX=1: DY=0
10100 X=PEEK(V+2)+DX: Y=PEEK(V+3)+DY
10110 POKEV+2,X: POKEV+3,Y
10120 RETURN
11000 REM ***        BURN          ***
11010 P(0)=116:P(1)=107:P(2)=226:P(3)=107
11020 P(4)=138:P(5)=147:P(6)=206:P(7)=147
11030 FOR I=0 TO 7: POKE V+1,P(I): NEXT
11040 POKE V+16,0: POKE V+28,255
11050 FOR I=0 TO 3:POKE2040+I,33:POKEV+39+I,2:NEXT
11060 POKE V+1,255:I=0
11070 FOR RU=1 TO 200: T=1-T
11080 POKE V+37,0+7*T: POKE V+38,7-7*T
11090 IF BU 100 THEN GOSUB 11140
11100 IF BU 50 THEN GOSUB 11160
11110 FOR D=0 TO 40:NEXT
11120 NEXT BU:POKE V+21,0
11130 FOR Y=0 TO 117:SYS MC,40,129,Y,0:NEXT:RETURN
11140 FOR J=0 TO 3: POKE V+2+J,P(2*J)-12
11150 POKE V+29,15: NEXT: RETURN
11160 FOR I=0 TO 3:POKE V+1+2*J,P(2*J+1)-20
11170 POKE V+23,15: NEXTI: RETURN
12000 REM ***        SKULL         ***
12010 FOR I=1 TO 12: FOR J=15 TO 24
12020 POKE SC+I*40+J,RND(0)*255: NEXTJ,I
12030 FOR I=14 TO 0 STEP -1: FOR J=0 TO 9
12040 MO=RND(0)*2+1: IF SK(I,J) 4 THEN 12080
12050 X=60+J*4: Y=40+I*4
12060 SYS MC,X,X+2,Y,MO
12070 SYS MC,X,X+2,Y+1,MO
12080 NEXT J,I
12090 RETURN
13000 REM ***      DRAW BOX         ***
13010 FOR I=Y1 TO Y2: FOR J=X1 TO X2
13020 POKE SC+I*40+J,134: POKE CO+I*40+J,C
13030 NEXT J,I
13040 FOR Y=Y1*8 TO Y2*8+7
13050 SYS MC,X1*4,X2*4+3,Y,C
13060 NEXT Y
13070 IF NS=1 THEN RETURN
13080 FOR Y=Y2*8+7 TO Y1*8 STEP-2
13090 SYS MC,X1*4,X2*4+3,Y,1
13100 NEXT Y
13110 RETURN
14000 REM ***      INITIALIZE       ***
14010 PRINT"⬛":POKE53280,11:POKE53281,14
14020 POKE53270,PEEK(53270)OR16:REM MULTICOLOUR MODE
14030 POKE53272,PEEK(53272)OR8:REM PUT SCREEN AT 8192
14040 POKE53265,PEEK(53265)OR32:REM SWITCH TO BITMAP
14050 DIM TR(33,15),DO(16,8),MO(37,10),SK(28,14)
14060 FORI=0TO20:FORJ=0TO15:READTR(I,J):NEXTJ,I
14070 FORI=0TO16:FORJ=0TO7:READMO(I,J):NEXTJ,I
14080 FORI=0TO14:FORJ=0TO9:READSK(I,J):NEXTJ,I
14090 V=53248:CO=1024:SC=55296:MC=49153:LI=0
14100 POKEV+8,100:POKEV+ 9,107:REM WINDOWS
14110 POKEV+10,230:POKEV+11,107
14120 POKEV+12,143:POKEV+13,147
14130 POKEV+14,210:POKEV+15,147
14140 POKEV+00,100:POKEV+01,60 :REM WITCH
14150 POKEV+02,50 :POKEV+03,80 :REM BAT
14160 FORI=0TO7:POKEV+39+I,1:NEXT:REM COLOURS
14170 FORI=4TO7:POKE2040+I,32:NEXT:REM POINTERS
14180 POKE2040,34:POKE2041, 35:POKEV+16,2
14190 POKEV+37,10:POKEV+38,8:POKEV+28,240:REM MC MODE
14200 FORI=0TO519:READ D:POKE2048+I,D:NEXT
14210 FOR I=0TO199:SYS MC,0,159,Y,0:NEXT
14220 RETURN
20000 REM      DATA FOR TREE
20010 DATA4,2,4,4,4,4,4,2,4,4,.,4,2,4,4,4,4
20020 DATA2,2,4,4,2,2,4,4,2,4,4,2,4,4,4,4
20030 DATA4,4,2,4,4,2,4,4,2,4,4,2,4,4,4,4
```

## APPLE IIe continued

```
7070  DATA BLOCK, 28, 60,118,120,  0,  0,1,2
7080  DATA BLOCK, 14, 40, 41, 80,  0,  0,1,2
7090  DATA BLOCK,105, 40,132, 80,  0,  0,1,2
7100  REM ROOFS ON TURRETS
7110  DATA TRIANGLE, 10, 39, 28,  0, 44, 39,5,5
7120  DATA TRIANGLE,103, 39,118,  0,136, 39,5,5
7130  REM BATTLEMENTS
7140  DATA BLOCK, 42, 45,103, 59,  0,  0,5,5
7150  DATA BLOCK, 49, 60, 55, 65,  0,  0,5,5
7160  DATA BLOCK, 63, 60, 69, 65,  0,  0,5,5
7170  DATA BLOCK, 77, 60, 83, 65,  0,  0,5,5
7180  DATA BLOCK, 91, 60, 97, 65,  0,  0,5,5
7190  REM WINDOWS & DOOR
7200  DATA BLOCK, 21, 43, 27, 51,  0,  0,0,3
7210  DATA BLOCK,119, 43,125, 51,  0,  0,0,3
7220  DATA BLOCK, 28, 60, 34, 70,  0,  0,0,3
7230  DATA BLOCK,112, 60,118, 70,  0,  0,0,3
7240  DATA BLOCK, 35, 90, 48,102,  0,  0,0,3
7250  DATA BLOCK, 98, 90,111,102,  0,  0,0,3
7260  DATA BLOCK, 63,105, 83,120,  0,  0,0,3
7270  DATA STOP,0,0,0,0,0,0,0,0
```

## COMMODORE 64 continued

```
20040 DATA4,4,2,4,4,4,4,2,4,2,4,4,2,4,4,2,4,4
20050 DATA4,4,4,2,4,2,4,2,2,2,4,4,2,4,2,4,2,4
20060 DATA4,4,4,4,2,4,4,4,4,2,4,4,2,2,4,4,4
20070 DATA4,4,4,4,2,4,4,4,2,4,4,2,2,4,4,4,4
20080 DATA4,4,4,2,2,4,4,4,2,4,2,2,4,4,4,4,4
20090 DATA4,4,2,4,2,2,2,2,2,4,2,4,4,4,2,4
20100 DATA4,4,2,4,2,2,2,2,2,4,2,4,4,4,4
20110 DATA4,4,4,2,4,4,2,2,2,2,4,2,4,4,4,4
20120 DATA4,4,4,4,4,4,2,2,2,2,4,4,4,4,4
20130 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4,4
20140 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4
20150 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4
20160 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4,4
20170 DATA4,4,4,4,4,2,2,2,2,4,4,4,4,4,4
20180 DATA4,4,4,4,4,4,2,2,2,2,4,4,4,4,4,4
20190 DATA4,4,4,4,4,2,2,2,2,2,4,4,4,4,4
20200 DATA4,4,4,4,2,2,2,2,2,2,2,4,4,4,4
20210 DATA4,,2,2,4,2,2,2,2,2,2,2,2,4,4
20220 REM  DATA FOR MOON
20230 DATA4,4,4,4,4,2,2,2
20240 DATA4,4,4,2,2,2,4,4
20250 DATA4,4,4,2,2,4,4,4
20260 DATA4,2,2,4,4,4,4
20270 DATA4,2,2,2,4,4,4,4
20280 DATA4,2,2,2,4,4,4,4
20290 DATA4,2,2,4,4,4,4,4
20300 DATA4,,2,4,2,4,2,4
20310 DATA2,,2,4,4,4,4,4
20320 DATA2,2,2,4,4,4,4,4
20330 DATA2,2,2,2,2,4,4
20340 DATA2,2,2,4,4,2,4,4
20350 DATA2,2,2,4,4,4,4,4
20360 DATA4,2,2,4,2,4,4,4
20370 DATA4,4,2,2,2,2,4,2
20380 DATA4,4,2,2,2,2,2,4
20390 DATA4,4,4,2,2,2,4,4
20400 REM  DATA FOR SKULL
20410 DATA 4,4,4,0,0,0,0,4,4,4
20420 DATA 4,0,0,0,0,0,0,0,0,4
20430 DATA 4,0,0,0,0,0,0,0,0,4
20440 DATA 0,0,0,0,0,0,0,0,0,0
20450 DATA 0,0,4,0,0,0,0,4,0,0
20460 DATA 0,0,4,4,0,0,4,4,0,0
20470 DATA 0,0,4,4,0,0,4,4,0,0
20480 DATA 0,0,0,0,0,0,0,0,0,0
20490 DATA 4,0,0,0,4,0,0,0,0,4
20500 DATA 4,2,0,0,4,4,0,0,0,4
20510 DATA 4,0,0,0,0,4,0,0,0,4
20520 DATA 4,1,4,0,0,0,0,1,4,1
20530 DATA 4,4,0,0,0,0,0,0,4,4
20540 DATA 1,4,0,4,4,4,4,0,4,4
20550 DATA 4,4,0,0,0,0,0,0,4,4
20560 REM  SPRITE DATA:WINDOW
20570 DATA ,255,255,0,86,84,0,250,188,0,90,148,0,250
20580 DATA 188,0,90,148,0,250,188,0,90,148,0,250,188,0
20590 DATA 90,148,0,250,188,0,90,148,0,250,188,0,90
20600 DATA 148,0,250,188,0,90,148,0,250,188,0,90,148,0
20610 DATA 250,188,0,90,148,0,255,255,0,000
20650 REM  FLAME
20660 DATA 0,192,0,0,1,64,1,0,0,1,104,0,5,156,0,15,196,0
20670 DATA 5,248,3,7,160,0,7,184,0,7,225,0,1,225,0,1
20680 DATA 231,0,192,232,112,48,104,112,40,232,112
20690 DATA 25,229,64,29,209,192,14,209,192,14,105
20700 DATA 192,7,184,184,1,174,184,255
20730 REM  WITCH
20740 DATA 0,0,0,0,0,0,0,0,14
20750 DATA 0,0,15,224,15,192,0,21,0,0,1,128,2,15,0,1
20760 DATA 255,0,1,250,128,4,249,128,0,60,192,160,128
20770 DATA 94,84,126,112,191,255,255,84,191,0
20780 DATA 120,7,0,0,50,0,0,60,0,0,112,0,0,112,0,000
20790 REM  BAT - WINGS UP
20800 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20810 DATA 56,119,0,219,129,219,129,0,60,0,0,24,0,0,36
20820 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20830 DATA 0,0,0,0,0,0,0
20840 REM  BAT - WINGS DOWN
20850 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20860 DATA 0,0,129,219,129,119,60,255,36,24,60,0,60,0
20870 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20880 DATA 0,0,0,0,0,000
```

# Glossary

**DATA**   A list of information that is required by a program. **DATA** can consist of words or numbers, or both together. A program is sent to the **DATA** with the instruction **READ**.

**FOR.....NEXT**   This is a sequence of commands that is used to make the computer repeat an operation a certain number of times. For example, the loop **FOR X=1 TO 5:PRINT 2*X:NEXT X** would cause the computer to print the two times table up to five.

**GOTO**   This statement tells the computer to go to the specified line, missing out any lines in-between. It is often used with **IF.....THEN** (see below) and is only operated if certain conditions are true. Be careful when using **GOTOs**, as it's easy to have the program jumping backward and forward so much that it is impossible to read.

**HGR**   This sets the high resolution graphics mode on the Apple.

**HPLOT**   This places a set of x, y coordinates on the Apple screen. If **HPLOT** is followed by **TO**, it draws a line from the last point plotted to the coordinates indicated. This works both horizontally and vertically.

**IF.....THEN**   This is used as a way of telling the computer to do something only when certain conditions are true. This instruction often looks something like this: **IF score=LE THEN WO=1.**

**INT**   **INT** is short for integer, and instructs the computer to make a whole number of a figure with decimal places in it. It is often used in conjunction with the **RND** command which instructs the computer to generate a random number (see below).

**LEFT$**   This instruction is used to copy part of a string, starting at the left-hand end. It is followed in brackets by the string name and the number of characters to be copied.

**LET**   This is one way of giving the computer information. In some programs there may be statements such as: **X=10**. This simply means that the number ten is stored under the label X. It is often clearer to write: **LET X=10**
The **LET** statement also gives rise to something that at first sight seems illogical, if not impossible. In many programs you will see things like: **LET X=X+1**
Of course, in mathematical terms **X** can't equal **X+1**. All this type of statement means is "increase the value of whatever is stored in **X** by one."

| | |
|---|---|
| **LIST** | This makes the computer display whatever program it has in its memory. You can **LIST** single lines, or parts of a program by following the **LIST** with appropriate line numbers. |
| **MID$** | This is used to copy the middle part of a string. It is followed in brackets by the string name, the start position, and the number of characters to be copied. |
| **PEEK** | This instruction looks at a particular memory location. It is often associated with **POKE**. |
| **PIXEL** | This represents a point on the grid in graphics mode. The number of pixels per screen is determined by the quality of the graphics, e.g. high or low resolution mode. |
| **POKE** | This stores numeric information in the computer's memory. It is often used for sound and places a binary number in a particular location. |
| **PRINT** | This tells the computer to display something on the screen. |
| **RIGHT$** | Similar to **LEFT$**, but copies the right-hand end of a string. |
| **RND** | This instruction makes the computer generate a random number. The precise instruction varies between different models of computer. |
| **shape tables** | These are used to define the shape of a graphic on the screen on the Apple IIe. They are stored in tables, so that they can be called up easily when required. |
| **sprite** | A sprite is a user-defined character on the Commodore 64 computer. It is a small area of the graphics screen which can be moved around and switched on and off with ease. A maximum of eight different sprites can be defined. |
| **STEP** | The **STEP** statement is always used following a **FOR....** statement. It indicates the amount that the variable should be changed for each operation. For example: **FOR X=0 TO 20 STEP 5: PRINT X: NEXT X** would mean that **X** would rise in steps of five, so that the computer would print 0, 5, 10, 15, 20. |
| **XDRAW** | This stands for "e**X**clusive **DRAW**." It is an instruction used on the Apple IIe. It combines what is being drawn "exclusively" with what is behind it. This means that it can be drawn over a background, and the background will return when the drawing is blanked out. It also means that when the same shape is "e**X**clusively **DRAW**n" in the same place twice, it disappears. |

# Index